Eliciting and Describing Users' Models of Computer Systems

Martina Angela Sasse

A thesis submitted to the
School of Computer Science,
Faculty of Science,
University of Birmingham
for the degree of
Doctor of Philosophy
(Computer Science)

April 1997

Abstract

The topic of this thesis is users' models: the representations users may form of the computer system which they are interacting with. It has been proposed that user interfaces which support the construction of appropriate users' models facilitate learning and use of computer systems. Users' models have been a topic of research in human-computer interaction (HCI) since 1984, but to date, no knowledge exists which could be applied by designers of computer systems. The aim of the thesis is to address this problem and contribute to the development of an integrated and applicable body of knowledge on users' models.

The thesis commences with an examination of the history and current state of the discipline of human-computer interaction to establish the context and determine the appropriate methods for conducting research on users' models. Since mental representations and mental models originate from the related disciplines of psychology and cognitive science, the review of the literature starts with an outline of the relevant theories, followed by a review of the theoretical and empirical work to date on users' model in HCI. The review concludes that more exploratory empirical work is required to obtain data from which evidence for, and descriptions of, users' models could be derived; however, suitable methods for eliciting and describing users' models have to be devised first.

The second part of the thesis describes a series of five observational studies of users interacting with application software. The studies employed different scenarios, ranging from traditional experimental-style scenarios, with users working through a series of tasks, to constructive interaction scenarios, where users interacted with a co-investigator playing the role of a learner or co-learner. All studies were recorded on video, transcribed and analysed. Advantages and drawbacks of the scenarios for eliciting users' models are identified and discussed. The analysis of the tapes and transcripts provides some evidence of users' models; the conclusions of the thesis provide an outline of how theories regarding users' models can be formulated, and tested, using the data collected.

Acknowledgements

The author conducted the research presented in this thesis with a scholarship from the School of Computer Science, University of Birmingham. The School purchased video equipment, without which the empirical work could not have been conducted, and provided travel funds to present papers on this work at a number of meetings in the UK and abroad. I would like to thank **Dr Peter Dodd** for setting up the scholarship and providing support for the research. Additional financial support, in form of a bursary and equipment, was provided by Philips Research Laboratories, Redhill. Very special thanks are due to my long-suffering supervisor, **Mr Dennis Parkyn**, for the advice, encouragement and support provided during the early years of the work.

The feedback provided by the examiners, **Dr Thomas Green** and **Professor Peter Hammond** on the Qualifying M.Sc. thesis, and **Professor James Alty, Dr Steven Draper**, and **Dr William Edmondson** on the thesis itself, was crucial for the academic progress of the work.

On a practical level, my fellow Ph.D. student and close friend, **Stephanie Keenan**, helped with the daunting - and at times seemingly insurmountable - task of transcribing approximately 150 hours of videotaped experimental sessions. She also acted as a co-experimenter in some of the sessions. The other co-experimenters were my fellow Ph.D. students **Ian Clarke** and **Ceri Hopkins**, and **Mark Flint**. The success of the spreadsheet study depended entirely on Ceri's performance as a contrived learner - something that is difficult to keep up over 20 sessions.

Finally, a tribute to the students at Birmingham University who were the users in the five studies - without them, the empirical work, which I regard to be the main contribution of this thesis, could not have been undertaken.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

The introductory chapter of the thesis serves two purposes:

(1) to provide background information on the context and motivation of the work described in the thesis;

(2) to offer a summary guide or map through the argument and research work presented in the remaining chapters.

1.1 Background

My own interest in human-computer interaction (HCI) in general, and users' models of computer systems in particular, dates back to the early 1980's. As a psychology undergraduate in Germany, I acquired my first PC and version 1.0 of the same word processing software which has been used 10 years later to write up this thesis. The purchase was motivated by the wish to reduce the time and effort spent on certain tasks related to my studies and the work which sustained me through my university years: producing course assignments, translations, and teaching materials for evening classes. Users of word processing software were enthusing about the amount of time saved on copy-typing and error correction; thus, I bought a computer expecting work to become less laborious and more fun. I wanted to spend more time and effort on the conceptual aspects of the document production task (planning of document structure, formulating text, etc.), and less time on procedures related to its presentational aspects (copy-typing, applying correction fluid, etc.).

The acquisition of a computer did result in lower consumption of correction fluid – but this seemed due to the difficulty in obtaining printouts to which the fluid could be applied, rather than the ease with which typing errors could be corrected. Much time was spent on consulting manuals to find out how to instruct the computer to perform familiar typewriter actions. The jargon in the manuals and the cryptic error messages added to the mystery of the system. Green (1990) later summarised this experience in the term *viscosity*: users struggling to identify the procedures required to make the system perform actions for a real-world task. The time and effort saved on repetitive and time-consuming procedures on the typewriter, such as copy-

typing, was exceeded by the time and effort spent on identifying and learning the procedures required to achieve a similar result using the word processor.

Not having Green's (1990) depth of understanding about the cause these problems, I attributed them to my lack of knowledge about computers. The next step therefore was to attend training course, where I encountered other users, many of them successful professionals and business people. They wanted to learn the system for the same reason: to cut down on tiresome routine tasks and gain time for the primary aspects of their work. Most of them also reported the same problems. The course taught procedures for the most frequently used functions of the software, which were easy enough to learn and recall. However, when trying to transfer those procedures to a slightly different task, or recover from an error, it was back to manuals and trial-and-error.

As a student of psychology specialising in industrial psychology, I was aware of theories of learning and instruction. The problem with the training programme and manuals seemed to be that, whilst providing an effective drill in the basic procedures required for a number of standard tasks, they did not furnish users with an understanding of how the system worked. Consequently, transferring learnt procedures to another task, or recovering from error, was therefore difficult to impossible.

Training courses and manuals for software products might therefore be improved by basing them on an instructional approach that went beyond the procedural drill, and provided new users with a conceptual understanding of the system. A conceptual understanding might take the form of a map or overview of the system, and explanation of what effect procedures had, rather than just showing how to perform them.

German academic psychology, a few *Gestalt* theoretical schools apart, is firmly based on positivist philosophy of science, and therefore takes a hard-line empirical approach. Only knowledge gained through experiments (preferably in a highly controlled environment) could provide a basis for developing a new training approach for computer software. Just because the benefits of concept-based learning had been demonstrated in science education and the training of process control plant operators, one could not assume that it would help with the understanding of

computer software. Before designing a new training approach, I had to demonstrate that a conceptual understanding of a software system did indeed provide benefits over a purely procedural drill and resulted in improved user performance. Thus, I devised a comparative study of procedural and conceptual training for new users of word processing software for my final-year psychology project. The German professors failed to see the potential scientific contribution of such a study at the time[1], but I was able to conduct it as an M.Sc. project at the Social and Applied Psychology Unit in Sheffield. The results confirmed the main hypothesis: even though there was little difference in performance on control tasks during the early stages of training, subjects in the conceptual condition performed significantly better on transfer tasks and made significantly less use of the help system and related external materials - such as crib cards - throughout (Sasse, 1986). A second, unanticipated conclusion from the project was that training was not the best way to improve the lot of computer users. Users often made errors as a result of plausible reasoning about the workings of the system, based on their observation of its behaviour. After exploring a number of software applications in depth, the conclusion was that improving the design of application software systems might be more effective than training (Sasse et al., 1988). User training may alleviate some of the problems caused by inappropriately designed software, but it is a case of treating the symptoms rather than the cause of users' problems. The potential gain – in terms improved user-system interaction – from shaping the tool seemed greater than the gain that could be obtained from shaping the users. This started an investigation into ways of improving the design of user interfaces to application software, one of the central concerns of HCI.

---

[1] The professor of occupational psychology, an expert on mentalistic learning (executing procedures mentally only – an approach mainly used by athletes to prepare for competition without the risks of wear and tear endemic in physical practise), suggested it would be more interesting to compare the effects of mentalistic *vs.* hands-on training for new computer users. It is difficult to envision an appropriate experiment (*"Imagine moving the cursor over the character and pressing the DELETE button"*), but given the recent rise of repetitive strain injury (RSI) in computer users, he may have had a point.

1.2 Map through the thesis

1.2.1 The research context

When the work for this thesis began in 1987, HCI was still struggling to establish itself as a topic in academic curricula. Early HCI research was conducted in both psychology and computer science departments. For a trained psychologist, remaining in a familiar environment to conduct this type of research might have been the obvious choice. The decision to opt for the unknown, but potentially more fertile, ground of a computer science department as home for this work was influenced by the debate about the *raison d'être* and direction of HCI research conducted at the time. A summary and discussion of that debate, and the conclusions for the research topic of this thesis, are presented in Chapter 2; the main motivation for conducting HCI research in a computer science environment was that *"design is where the action is"* Newell & Card (1985).

The route to changing the way in which computer systems are designed is via the minds of the people who design them. A computer science department offered the chance to "go native" and work with designers to computer systems and user interfaces, learn about their language, culture, and view of the design process. Thus, I took the opposite direction on the ethnomethodology trail blazed by Suchman (1987) and others, who proposed that designers or their agents should use anthropological methods to gain an in-depth understanding of users' culture, language and task as a basis for system design. In addition to helping me to gain an understanding of how designers work and think, the move provided a welcome opportunity to build up my own knowledge of the "C" part of HCI.

Even though this sojourn was not without friction[2], I still feel strongly today that such an understanding is an essential first step to getting HCI knowledge and practice into the minds and hands of designers. In addition, staying close to the

---

[2] Not all computer scientists appreciated HCI researchers invading "their" territory. The objection I most frequently encountered during my time as a PhD student was the one raised Newell & Card (1985): that HCI was "too soft" to be incorporated into computer science or system design. Members of the departmental hacking fraternity referred to people without a computing or science degree as "basket-weavers". I was quite fond of this label, since ironically, most work in computing, particularly programming, is *craft*- rather than *science*-based (see 2.2.1).

action in system design seemed to be the best safeguard against conducting HCI research whose results come too late to be incorporated into the design of next generation of computing technology (Newell & Card; 1985).

1.2.2 The research topic

The choice of mental models as a basis for designing more usable software systems was, in many ways, a logical progression from my the previous investigations of procedural and conceptual training (Sasse, 1986; 1988), but the inspiration to pursue this particular topic can be traced to two presentations at the 1986 Annual Conference of the British Psychological Society.

At the conference, Philip Johnson-Laird received the President's Award for his theory of mental models. The theory (Johnson-Laird, 1983) states that people construct working cognitive representations of phenomena they interact with from procedural building blocks, via the application of a set of rules. These representations – *mental models* – therefore accommodate both procedures and conceptual structures. The theory has the potential to explain observations from instructional practise. One of those observations, demonstrated by Johnson-Laird himself in a number of experiments, is that people can process identical information differently, and show differences in performance on the same task as a result. Differences in processing of procedural building blocks lead to the construction of more or less appropriate mental models, which in turn lead to differences in performance on a task. The differences in processing are linked to the extent to which incoming information is associated with already existing knowledge. One of the main advantages of Johnson-Laird's (1983) theory, as discussed in more detail in 3.2, is that it offers a description of what mental models look like and how they are constructed. It is therefore possible to derive hypotheses of how the construction process might be influenced to facilitate the construction of appropriate models. The empirical investigations on which the theory is based have, however, been limited to very a specific task domain, that of syllogistic reasoning.

Norman & Draper (1986) applied the notion of mental models to HCI, to bridge the gap between designers and users of software systems. A designer will have formed a working model of a system she has designed. Users, learning to operate that

system through interaction and/or some form of instruction, will form a working model which is different from the designer's. The fact that the models are different is not necessarily a problem; the problem arises when a user's model is inappropriate. The observable effects of users building a model which fails to support their interaction with the system range from selection of inefficient procedures for completing tasks to severe disruption or breakdown of the interaction (errors and problems with error recovery, inability to identify requested functionality). Norman & Draper (1986) proposed that designers could facilitate the construction of appropriate users' models of a system through the design of its user interface. The designer can decide on an appropriate model and structuring the presentation of information and user-system interaction accordingly. The idea of *conceptual design* approach is appealing, but – as discussed in more detail in Chapter 4 – it was presented as a concept only, without sufficient procedural support in the form of guidelines or examples to be immediately applicable[3].

The research project described in this thesis was undertaken with the aim of generating HCI knowledge which could contribute to putting the conceptual design approach into design practise, i.e. the minds and hands of designers.

1.2.3 The results of the literature review

Any research undertaking commences with a critical review of the relevant literature on the topic. The review of the literature on users' models in HCI (in chapter 4) is divided into two parts: (1) a review of the theories of users' models, and propositions how these theories could be applied; and (2) a review of empirical investigations of users' models and the knowledge gained from them.

The analysis of existing theoretical contributions uncovered a range of theories on the nature and working of users' models. Some of these theories are overlapping or complementary, but there is also a fair amount of contradiction. This epistemic state

---

[3] The discovery that it might not be straightforward to put this idea into practise came early days of in my PhD research. All excited by the idea of conceptual design presented in Norman & Draper (1986), I tried to convert a designer and fellow member of the department to this approach. He thought about it for a while and then said *"All right – I'll try it for my next system. Just tell me what I have to do to get the right conceptual model."*

is reflected in the terminological muddle presented in the literature. Similarly, the survey of empirical studies revealed fragmented and contradictory evidence.

The conclusion from the review of theories was that, whilst there was general agreement among the authors that users' models do exist and influence user-system interaction, there was no agreement about the actual form and working of those models. This state can be attributed to a lack of the most basic substantive knowledge about users' models: convincing evidence that they exist, and testable descriptions of how they work. The requirement for this substantive knowledge was confirmed in a survey of mental models in HCI by Carroll & Olson (1988). Their 11 important research needs - essentially a research agenda for producing substantive knowledge about mental models - inspired one of the two major strands of my own research agenda: to provide evidence and descriptions of users' models. The second major strand was prompted by the review of the empirical studies undertaken up to that point in time: to provide valid methods for producing such evidence and descriptions.

1.2.4 The research method

The review of the empirical studies of users' models raised a number of methodological concerns. As discussed in Chapter 5, users' models are internalised cognitive representations, and therefore not directly observable. The existence of such models is therefore inferred from a range of observable behaviours displayed by the users. A description of users' models will be a researcher's externalised conceptualisation of users' internalised models. The methodological concerns raised in Chapters 4 & 5 are (a) which observable behaviours displayed by users can be used as evidence for the existence of internalised models and basis for conceptualisation; and (b) the circumstances under which users' observable behaviours – the data – can be elicited.

The aim of the research undertaking was to find evidence of users models and generate descriptions of them. The research method chosen to address this question was based on three major concerns:

(1) Given that there is no universally accepted research instrumentarium in the emerging discipline of HCI (see 2.3), the research method should be chosen to suit the phenomenon under investigation.

(2) Since existing HCI knowledge about users' models was fragmented, with a number of theories which are neither complementary nor competing, an experimental approach testing hypotheses did not seem the most appropriate strategy (see 4.4). Rather, an exploratory approach was required in which user behaviour would be screened for any evidence of user's models, using any method of description which seemed suitable.

(3) Since the relationship between observable user behaviours and users' internal representations had not been established sufficiently to rely on one particular type of behaviour as indicator for users' models, it seemed prudent to capture and analyse a range of observable user behaviours (see Chapter 5).

A core contribution of the research work described was to devise a range of scenarios, and determine their potential for eliciting users models. To ensure that the results would be valid and applicable, real software systems, tasks and users were investigated. Detailed reports of the use of each of the scenarios, and observations made on users' models, are presented in Chapters 6-10. The aim was to capture a substantial amount and range of relevant data, which could be used for HCI research on users' models.

1.2.5 The results of the empirical studies

A summary of the methodological and substantive knowledge collected in the studies is presented in Chapter 11. On the methodological side, the studies demonstrated that constructive interaction scenarios are most suitable for exploratory studies, i.e. eliciting users' models. They require careful planning and execution, and the effort involved in preparing and analysing the data collected is substantial. Structured scenarios which aim to test users' knowledge require less time to prepare, conduct and analyse, but have to be carefully targeted otherwise they may yield less information about the content or form of the model. Most of the substantive knowledge is, at present, still contained in the data collected. Ideally,

the videotapes should accompany the thesis. Since this is not feasible, transcripts of all experimental sessions are provided in a separate volume. Examples from the studies illustrate how descriptions of users' models can be derived to form the basis for theories which can then be tested against the data. It is argued that this approach to formulating and testing theories of users' models - adapted from the social sciences - may offer a long-term perspective of integrating the currently diverse knowledge on users' models into an applicable body of HCI knowledge.

2 The Research Discipline of HCI

The purpose of this chapter is to establish the disciplinary background and context for the research undertaking presented in this thesis. HCI has a multidisciplinary background and a relatively short history as an independent research discipline. Its *raison d'être* is to provide knowledge and methods for the design of usable computer systems; hence, the effectiveness of the discipline will ultimately be judged by the usability of computer systems which are being produced. This means that to succeed, the discipline not only needs to establish a body of knowledge, but to ensure that this knowledge is applied successfully by those who design and build computer systems. There is currently no consensus in the discipline as to the research strategy and methods by which this can be achieved; therefore, a decision about research strategy and methods needs to be made by the individual researcher for the specific topic under investigation.

2.1 Brief history of HCI

Over the last three decades, computers have become an integral part of many people's jobs. Most organisations have introduced computerised systems, expecting to improve productivity by speeding up work cycles, reducing the number of repetitive tasks, and increasing employee flexibility. More often than not, however, the introduction of computer systems did not meet these expectations. Even in organisations which planned the introduction of their systems carefully, many users of the new technology encountered problems. Whilst technology suppliers initially attributed many of these problems to technophobia or resistance to change, user organisations often discovered that employees had been left to struggle with systems which were difficult to learn, difficult to use, and did not provide appropriate support for their tasks.

Before the start of the microcomputer revolution of the seventies, most computer systems were designed by computer professionals and used by computer professionals. With the spread of personal computers (PCs), the user population expanded to include people who were experts at their own jobs and not computer professionals. This new generation of users had neither the time nor the motivation

to acquire extensive background knowledge in electronics and programming: they wanted to use the computer as a tool to help them do their work more effectively and efficiently. The discrepancy between users' expectations and goals, and the usually frustrating experiences resulting from attempts to use computer systems, set the background for the emergence of a new research discipline: human-computer interaction (HCI).

The speed of hardware development and increasing competition amongst suppliers meant that the cost of hardware decreased in real terms. Perceived user requirements began to replace hardware constraints as the dominant influence on the software development process. Users' buying decisions shifted from being hardware-led – select hardware, then find or commission suitable software – to being software-led – select a suitable software package, then buy the hardware needed to run it (Grudin, 1991). Terms such as *user friendly, easy to learn* and *easy to use* are important marketing tools today[4], and software companies attach much importance to the *look and feel* of the user interfaces of their products[5]. The user interface has become a major determinant of the success or failure of computer systems (Gaines & Shaw, 1986). With both industry and users looking to HCI researchers and practitioners for guidance, HCI is very much an applied discipline, and its *raison d'être* is to provide knowledge and tools for the design of usable computer systems.

HCI in general, and the design of "good" user interfaces in particular, has been an extraordinarily popular area of research over the last 10 years. The subject has attracted researchers and practitioners from mainly three established disciplines: psychology, computer science, and human factors or ergonomics. These disciplines have contributed to HCI from its early days. More recently, there has been research input from the fields of sociology, anthropology, linguistics, organisational behaviour, fine arts and various design disciplines. The number of publications in the field has soared, journals and book series specialising in HCI have appeared.

---

[4] *Viz.* the – unsuccessful – attempt of one manufacturer (Amstrad) to trademark the term *user friendly* for its products in the early nineties.

[5] *Viz.* the – unsuccessful – attempt of one software company (Apple) to sue a competitor (Microsoft) over copying the *look and feel* of the user interface to their operating system.

The professional bodies of relevant disciplines have established specialist groups. HCI conferences are held regularly at national and international levels. HCI courses and options are taught at higher education establishments all over the world. Whilst there can be little doubt that HCI has established itself as a discipline, there has been some argument over the nature of the discipline, and how its research should be conducted and communicated. The following section examines the different positions with a view to establishing the disciplinary and methodological background for the specific research undertaking presented in this thesis.

2.2 Defining the nature of the discipline

2.2.1 HCI as a science

The view of HCI as a science in the traditional sense was put forward by Newell & Card (1985). The authors' position was based on their concern that psychology – a "soft" science contributing to the "H" part – was about to be pushed out of HCI by the "hard", formal contributions of the "C" disciplines. To counteract this development, Newell & Card (1985) proposed a programme to transform relevant psychological knowledge into a "hard" science, by providing *engineering-style theories of the user* which could be applied by designers of new technologies. Clearly, the authors – both researchers with a long-standing interest and scientific record in psychology and among the earliest contributors to HCI – were worried about the validity and applicability of psychological research in HCI.

During the early days of HCI, there had been much interest in the psychological view of usability problems, and plenty of opportunity to conduct research and to publish the results. Designers looking for guidance and advice on how to design more usable systems, however, were unable to apply this research to the design of novel products. Most experimental results were obtained under conditions too specific to generalise, and originated from work with obsolete technology. At the same time, available design guidelines were too high-level to be applied to specific design decisions, and often contained conflicting recommendations.

Newell & Card (1985) detected increasing scepticism amongst designers towards psychological contributions to HCI, and feared that they would look for more

formal, scientific methods (such as formal specifications) as a basis for designing user interfaces and structuring user-system interaction. As a result, psychological concerns would be excluded from the design process. To safeguard psychology's future in HCI, the authors proposed a programme to turn relevant psychological research into a "hard" science, capable of supporting decision-making in the design process. This could be achieved by incorporating psychological knowledge in models and design tools used by designers. The framework for the construction of such models and tools would be an *engineering-style theory* of user behaviour, based on:

(1) *task analysis*: symbolic descriptions of tasks from which user behaviour can be predicted;

(2) *calculation*: prediction of user behaviour through explicit operations on mathematical models;

(3) *approximation*: acceptance that such calculations can only be approximate because of the complexity of human behaviour.

It the notion of approximation which makes this approach an *engineering-style* rather than a purely scientific one. An example of an engineering-style theory available at the time were the authors' own *GOMS* and *Keystroke-level* models (Card, Moran & Newell, 1983). The authors conceded that these models had been criticised – by designers – as being:

- too low-level;
- too limited in scope;
- focused on outdated technology;
- too difficult to apply.

Therefore, their own models suffer from very similar shortcomings to those which Newell & Card (1985) diagnosed in the psychological contribution to HCI in general. They proceed to outline how these limitations could be overcome in subsequent engineering-style models.

According to Carroll & Campbell (1986), however, their programme failed to redress those shortcomings. Jack Carroll and a team of fellow researchers have contributed

a large number of exploratory, qualitative studies of user-system interaction (e.g. Carroll et al., 1985; Aaronson & Carroll, 1987). This line of research, they argue, had produced valuable applicable knowledge about user-system interaction. Consequently, they dismiss Newell & Card's (1985) call to harden psychological science in HCI by imposing severe methodological strictures as

> *"... ritual scientific flagellation ... [and] a source of intellectual overhead periodically encountered by scientists ..." (p. 227).*

Carroll & Campbell (1986) denounce the labelling of the different approaches as "hard" and "soft" as a semantic sleight of hand, which places anyone conducting HCI research outside the engineering-style framework in the unfortunate position of advocating "soft science" – a chip on the shoulder of many psychologists, often accused by other scientists of practising anything but proper, "hard" science. This is rather ironic, since most work in computer science is actually "soft" by Newell & Card's (1985) standards – based on informal observation and exploratory programming, rather than scientific experiments and inference[6]. The survival of psychology in HCI may therefore not depend on its ability to deliver formal descriptions and approximate calculations. Quite to the contrary: restricting the psychological contribution to knowledge in this form would mean discarding a wealth of applicable information about users, and therefore reduce psychological input to HCI. Carroll & Campbell (1986) suggest that, rather than reducing the psychological contribution to HCI to quantitative measurements and calculations, psychologists should ensure that the value of explanatory, conceptual HCI research (in the vein of the studies performed at the IBM User Interface Institute throughout the eighties under Jack Carroll) is recognised and applied.

---

[6] They could call on evidence from computer scientists such as Abelson et al. (1985), who state that: *"Underlying our approach to this subject is a conviction that computer science is not a science [...] [it] might at best be called* procedural epistemology *– the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects. Mathematics provides a framework for dealing precisely with notions of 'what is'. Computation provides a framework for dealing precisely with notions of 'how to''. p. XVI*

Both sides in this argument see HCI as a science, and agree that the future of the discipline depends on its ability to deliver theories and tools which support designers in their efforts to produce usable systems. The disagreements are over

- the types of theories and models that are applicable to design;
- the type of data to be used as a basis for theories and tools;
- the method and context of enquiry used to procure those data.

According to Newell & Card (1986), theories have to be technical, i.e.

> *"... formal, manipulation-oriented, idealised approximations, whether they are symbolic, structural, or numerical in form and whether they use rules, theorems, grammars, simulations, or formulae." (p. 259)*

Empirical HCI research should follow the traditional science approach and collect quantitative data as a basis for building and evaluating mathematical models. Carroll & Campbell (1986), on the other hand, advocate conceptually deep, explanatory theories of how humans interact with machines. Since HCI researchers, like psychologists, are in no position to observe mental processes directly, it is necessary to collect *"all the information we can get."* (See Table 1 for a summary.)

Authors	Newell & Card (1985, 1986)	Carroll & Campbell (1986)
Types of theories	engineering-style predictive	conceptual explanatory
Models employed	idealised approximations	none specified
Context of enquiry	traditional scientific experiments	observation of real-world tasks
Data collected	quantitative performance data such as times, errors	qualitative any observable behaviour, incl. verbal protocols

Table 1: "Hard" vs. "Soft" science in HCI

In their response to this counter-proposal, Newell & Card (1986) point out that they do not reject qualitative research as such. Both quantitative and qualitative research have a place in HCI, since observations and detailed descriptions of phenomena provide the foundations of any research activity. Their central criticism of the route

proposed by Carroll & Campbell (1986) is that it does not progress beyond the informal, qualitative stage:

> *"... it is necessary to move beyond informal description to technical theories of mechanism. Our fear is that, without taking the next step, the account attempted of behavioural phenomena can degenerate into woolly verbal argument protected by ad hoc adjustment – to anecdotal science." (p. 260)*

There certainly is plenty of anecdotal science to be found in HCI, and the discipline has suffered its fair share of woolly verbal arguments in the past, never resolved because the differing views were not formulated clearly enough and put to the test. This state of affairs does result when recommendations are solely based on explanation derived by intuition, and justified with selected clips from subjects' behavioural or verbal protocols. The central weakness of Carroll & Campbell's (1986) argument was that they gave no indication of how their research should progress from those informal descriptions – a shortcoming they subsequently addressed by formulating their view of HCI as a design science (Carroll & Campbell, 1989; see 2.2.2).

On the other hand, the approach advocated by Newell & Card (1986; 1985) harbours its own dangers for the discipline. Chief amongst these are the temptation to formalise phenomena before they are sufficiently well understood, and abstracting complex phenomena (such as human reasoning) into simplistic, mechanistic models. As Green et al. (1988) point out:

> *"... when the material to be formalised fails to conform to the expected structure, formal systems encourage ingenuity in squeezing difficult material into shape rather than rethinking the system and developing a better approach." (p. 3)*

Application of quantitative methods to behavioural and cognitive phenomena which have not been sufficiently well described and understood can lead to the *garbage in, garbage out* problem well known in computer science. Carroll & Campbell (1986) are right to point out that the imposition of methodological strictures alone will not make HCI more creditable or applicable. It could have the opposite effect: a discipline can be strangled by methodologies strictures which become more important than the theories they are supposed to serve. Psychologists' over-zealous

striving to be accepted as scientists reduced psychology to a rat-and-maze science for a significant part of its history. If engineering-style theories of the user restrict HCI to collecting observable, quantitative data for mechanistic models, without trying to explain the underlying cognitive concepts, HCI would be in danger of repeating this history, rather than learning from it.

2.2.2 HCI as a design science

Carroll & Campbell (1989) formulated *artifact theory* as an alternative research strategy for HCI to Newell & Card's (1985; 1986) engineering-style theories. Artifact theory outlines how HCI could progress from an informal, qualitative approach to a science discipline without developing mechanistic models of users.

The premise of artifact theory is that in HCI, research issues tend to be articulated and systematically studied only after they have appeared in a system or *artifact*. Whilst Newell & Card (1985) describe this situation as the problem of HCI science lagging behind computing technology, Carroll & Campbell (1989) take a more optimistic view by arguing that these artifacts perform the same function in HCI as theories in traditional scientific disciplines. Through interpretation and evaluation of these artifacts in the context of real use, HCI research will eventually formulate more conventionally expressed theories of user-system interaction.

Carroll & Campbell (1989) justify their departure from the traditional route of scientific enquiry with three main arguments:

(1) Because of its interdisciplinary nature, HCI needs to develop its own approach and its own set of tools. Straightforward application of traditional theories and methods from academic psychology had proved to be *"unilluminating and distortive"*, and were designed to maintain the stifling division between contributions from the "H" and "C" sides of HCI, rather than furthering integration.

(2) The epistemic *zeitgeist* emphasises the importance of contextual issues to produce valid and applicable research results. A forward-looking discipline should broaden the focus of enquiry, rather than attempt to squeeze complex behaviour into mechanistic models and restrict

observation to measurable performance. Generalisation and abstraction in the early stages of enquiry are not appropriate if we want to assess the impact of individual and situational factors. HCI can only produce valid and applicable results by acknowledging the individuality of users and their tasks, and the complexity arising from the interaction of different characteristics.

(3) If real users and real tasks are its central concern, HCI requires a greater degree of involvement from researchers than most traditional scientific disciplines. Therefore, it cannot limit its investigative instrumentarium to the confines of theory, hypothesis and test; other, more appropriate methods of conducting HCI research have to be found.

Indeed, an increasing number of active HCI researchers seem to subscribe to these arguments and reject the traditional approach of scientific enquiry. Monk & Wright (1991) observed that more recent work in HCI is much less likely to be conducted in the form of traditional hypothesis-testing experiments. They attribute this to the proven lack of generality of most HCI results gained through traditional experiments. Notable exceptions are problems belonging to the lower level domains of motor control, perception and lower cognitive mechanisms. These domains have been thoroughly researched under the aegis of traditional psychology, and their theories, models and results have been shown to apply to comparable tasks in the domain of HCI.

Monk & Wright (1991) declare design science as the most promising direction to pursue for researchers and designers investigating higher-level HCI phenomena. They advocate the use of *observation as a method* to state generalisable truths about user behaviour. The method is called *observation-invention pairs:* the invention (usually a new system feature or artifact) serves to clarify the nature of the observation that inspired it, and at the same time justifies the importance of the observation. (If the observation had been unimportant, it would not have warranted the invention of a new design feature. The observation evaluates whether the invention serves its purpose, and may give rise to further inventions.) Observation-invention pairs:

(1) emphasise the relationship between *user behaviour* and *artifact*, without making any detailed psychological claims about that relationship;

(2) describe the observed relationship at level which is not too detailed, but not too general, either, using *folk psychology* vocabulary accessible to designers;

(3) translate data from observations directly to ideas for design.

The effectiveness of the invention and the accuracy of the observation can only be evaluated by face validity and in accordance with the designer's intuition.

Monk & Wright's (1991) caution that

> *"Some might argue that such an approach is unscientific ..." (p. 215)*

is a formidable understatement even by British standards. The suggestion that a method which:

- is based on a circular definition (observation-invention)
- is expressed in *folk terminology*
- evaluates its results on the basis of *face validity* and *intuition*

could be an instrument of science is bound to be received with howls of derision by HCI researchers from the traditional science camp. Presented in this way, artifact theory could be accused of performing a semantic sleight of hand to claim a science base for what is essentially craft-based activity (see 2.2.3).

Nevertheless, Carroll & Campbell (1986, 1989) would argue that, with artifact theory and observation-invention pairs, HCI has finally broken the stranglehold of methodological restrictions of traditional psychology, and evolved into a discipline in its own right, making useful contributions to the design of computer systems. For Newell & Card (1985, 1986), this development would confirm their worst fears: HCI abandoning the possibility of a sound science base and heading for a future as an anecdotal science, to be ultimately replaced by formal approaches to user interface design.

2.2.3 HCI as an engineering discipline

The view of HCI as an *engineering discipline* has been formulated by representatives of the Human Factors (HF) tradition. Long & Dowell (1989) define HCI as

> *"...the design of humans and computers interacting to perform work effectively." (p. 13)*

They decompose the discipline into the complementary problems of:

- the design of humans interacting with computers (the domain of human factors); and
- the design of computers interacting with humans (the domain of software engineering).

Even though the terms are very similar, the view of HCI as an *engineering discipline* is different from the *engineering-style theories* advocated by Newell & Card (1985). HCI as an engineering discipline views the user as a component of the overall system which, like the computer, is to be shaped (e.g. through training and instruction), whereas the engineering-style theories of Newell & Card (1985) use scientific methods to produce

	Craft	Science	Engineering
HCI problem addressed by	implementation & evaluation	hypothesis & test	specification & implementation
type of knowledge	heuristics	theories, models, laws, hypotheses	engineering principles
knowledge acquired through	practice (experience and training)	scientific enquiry (deduction/ induction)	induction from craft knowledge deduction from scientific knowledge
cost of knowledge acquisition	low	high (induction) low (deduction)	high

Table 2: HCI as a craft, science and engineering discipline (after Long & Dowell, 1989)

models of the user as a basis for design. Long & Dowell (1989) base their argument for HCI as an engineering discipline by contrasting its prospects with those of HCI as a craft and a science (see Table 2 for an overview). HCI as a craft discipline typically solves problems by *implementation* and *evaluation*. Craft knowledge is *implicit* and *informal* (in the form of *heuristics*), and can be acquired *experientially* (by practice and example). Whilst the craft-based approach might successfully tackle individual design problems, Long & Dowell (1989) deem it to be ineffective as a research approach because:

(1) it is not operational and its knowledge is therefore not testable;

(2) it cannot guarantee the effectiveness of its solutions;

(3) its knowledge cannot be generalised.

The cost of acquiring craft knowledge is deemed to be acceptable.

HCI as a science solves problems by hypothesis and test. Scientific knowledge is acquired by processes of *deduction* (from theories) or *induction* (from observable behaviour).

According to the authors, HCI as an applied science is not effective because:

(1) it cannot be directly applied, since its knowledge is explanatory and predictive, rather than prescriptive;

(2) it does not make any statements about performance.

Since extensive data collection is required as a basis for induction, the cost of acquiring scientific knowledge will originally be high. Once guidelines can be generated by deduction from existing knowledge, the cost becomes low.

HCI as an engineering discipline would solve problems by *specifying* and *implementing* designs. Engineering knowledge is *codified, formal* and *operational,* and presented in form of *engineering principles.* These principles enable designs to be prescriptively specified for artifacts or systems which demonstrate a prescribed and assured performance when implemented. Since HCI can be decomposed into the domains of software engineering and human factors, both disciplines would construct their own engineering principles. Software engineering is assumed to be

close to establishing such principles, whilst human factors is currently not in a position to formulate them.

The programme for developing human factors principles is outlined in a second paper (Dowell & Long, 1989). A prerequisite for the formulation of engineering principles is a *conception* – a unitary and consensual view of the discipline. The conception proposed by the authors is based on two assumptions: firstly, that *human behaviour is essentially deterministic* (at least when interacting with a computer system); secondly, that the *effectiveness of interaction* (between a human and a computer) can be expressed in terms of *performance*[7]. Once the conception has been formulated, knowledge can be developed top-down from scientific theory by deduction, or bottom-up from craft knowledge by induction. A major research programme would be required to formulate the conception and develop the principles, so the cost of acquiring such knowledge would be high.

The proposal is an intriguing one, but one has to raise the question if and when the consensus necessary for formulating the conception can be achieved. The view that users' behaviour is essentially deterministic, and effectiveness of user-system interaction is solely evaluated in terms of performance, will be challenged by many researchers and practitioners in the discipline – certainly Carroll & Campbell (1986; 1989). It is interesting to note that human factors' sister discipline, software engineering, has encountered the same problem:

> *"The application of engineering principles is not straightforward; compared to most products, software engineering is in essence much more abstract ... As with management and office work, discussions of productivity run into measurement problems; the most common measure, 'lines of code', takes little account of programming style or language characteristics." (p. 171)*

This quote from Angell & Smithson (1991) not only illustrates the problem with achieving a disciplinary consensus, but reveals that establishing engineering principles in HCI as envisaged by Dowell & Long (1989) will be the ambitious undertaking they admit it is, twice over. Their assumption that human factors' sister

---

[7] On this particular point, the engineering principles of Dowell & Long (1989) and the engineering-style theories of Newell & Card (1985) concur.

discipline, software engineering, is close to establishing engineering principles, is over-optimistic. This seems to be a second incarnation of the semantic problem identified by Carroll & Campbell (1986) in the soft *vs.* hard science debate in 2.2.1. The problem is the – often implicit – assumption that, because research on the "C" side of HCI is carried out in disciplines named computer *science* and software *engineering*, all research in these areas follows strict science and engineering procedures and yields "hard" results. This assumption is incorrect. Calls for more rigour in research on the "H" side of HCI in order to meet standards on the "C" side may therefore indeed be an act of ritualistic self-flagellation, caused by an over-optimistic assessment of the state of knowledge and research methods in the sister discipline.

2.3 Conclusions

The reason for examining history and nature of the discipline of HCI was to establish the disciplinary context for the research work presented in this thesis. The purpose of any research undertaking is to contribute to the advancement of knowledge. The starting point for any advancement is to establish relevant existing knowledge, and the methods through which new knowledge can be acquired. Whilst there can be no doubt that there is a requirement to advance HCI knowledge, the result of HCI's multidisciplinary background and relatively short tradition as a research discipline is a lack of consensus about boundaries of established knowledge and suitable methods for generating new knowledge. The review has examined three different views of how the research should be carried out in order to advance the discipline:

(1) HCI as a traditional science tempered by approximation, providing engineering-style theories and tools for designers (Newell & Card, 1985; 1986).

(2) HCI as a design science, developing a craft-based approach and new research methods to evaluate existing systems in their intended and tasks context, using the results to inform designers for the next generation of systems (Carroll & Campbell, 1989).

(3) HCI as an engineering discipline, accumulating knowledge and formulating engineering principles which form the basis for design and evaluation (Dowell & Long, 1989).

The authors of all three proposals argue that their particular approach and research methods will generate the knowledge required to design more usable systems. There are, however, also arguments why pursuing either of these approaches exclusively may not produce the integrated body of applicable knowledge which the discipline needs to establish.

(1) HCI as an approximate science requires that HCI knowledge is expressed in engineering-style models of the user. It is certainly possible to formulate such models for those parts of HCI where we have existing knowledge from psychology to draw on: phenomena relating to motor control, perception and lower-level cognition are well understood and should be applicable to HCI (Manktelow & Jones, 1987; Monk & Wright, 1991). Phenomena which are part of higher-level cognition, however, are not as well understood; in particular, the relationship between higher-level cognition and performance is not well established. Expressing knowledge about higher-level cognition in HCI as engineering-style models would, therefore, be a case of premature formalisation where a new approach may be required (Green et al., 1988). Another point against expressing knowledge in the form of scientific models and approximate calculations is, despite Newell & Card's (1985) statement that designers require knowledge in this form, Bellotti (1988) reports that science-based modelling techniques such as Command Language Grammar (CLG – Moran, 1981), Cognitive Complexity Theory (CCT – Kieras & Poulson, 1985), and Task-Action Grammar (TAG – Payne & Green, 1986) have not been taken up by designers. These methods seems to fail designers' cost/benefit test: the overhead required to comprehend and apply such methods is seen as too large. Many designers feel that the usability problems would also be identified through less complex and time-consuming methods.

(2) HCI as a design science positions itself closely to the work of designers, aiming to provide knowledge which is relevant and immediately applicable. It does not, however, offer a convincing strategy for accumulating and integrating the craft knowledge generated by distributed teams of researchers into a body of HCI knowledge. The argument that HCI knowledge can be accumulated in the form of designed systems must be rejected since we know that the design of a system is influenced by considerations other than usability: economic (e.g. marketing) and legal (e.g. copyright) considerations play a substantial role in the conception, design and development of systems. Whilst arguing strongly for the use of HCI-specific methods rather than traditional scientific ones, the design science camp has not offered much in the way of methods. Observation-invention pairs, which are generating only folk-psychology descriptions and ideas for design, are throwing the analytical baby out with the experimental bathwater.

(3) HCI knowledge in the form of engineering principles may appear to be the halfway house between the traditional scientific and the design science approach, offering HCI knowledge in a form which is sound and accessible at the same time. The construction of these engineering principles, however, is based on a conception requiring a consensus within the discipline which does not exist at present.

The lack of a single agreed research strategy for the discipline leaves an individual researcher planning a specific research undertaking out on a limb. The analysis of the different approaches in this chapter has, nevertheless, yielded a number of pointers as to why and how HCI research should be planned and conducted:

(1) HCI knowledge needs to be made accessible and applicable to designers in order to achieve its intended effect. Designers are users of HCI knowledge; therefore it has to presented in format which supports them in the task they are trying to perform. This means it has to be translated into accessible guidelines, efficient methods, or incorporated into tools.

(2) HCI needs to build a body of knowledge; therefore, any research undertaking in HCI should aim to contribute to the accumulation of HCI knowledge.

(3) A body of knowledge cannot just be accumulated; HCI knowledge needs to be integrated and expressed in a unified terminology.

(4) Existing knowledge from related disciplines needs to be evaluated and integrated where appropriate. Without such a process, relevant knowledge from an existing discipline may be ignored and needlessly replicated in HCI, or existing knowledge may be transferred which does not apply to HCI. Prior to a research undertaking, the researcher should determine

- what related knowledge from other disciplines exists,
- whether it can be adopted in its existing form, or
- whether an HCI-specific version needs to be constructed.

(5) The process described in (3) may lead to the identification of conflicting theories from different disciplines; resolving which of these applies to HCI is also part of the integration process.

(6) Existing psychological knowledge on motor control, perception and lower-level cognition can be applied in HCI and expressed in the form of approximate scientific models (Manktelow & Jones, 1987; Monk & Wright, 1991). The applicability of existing theories and models of higher-level cognition, on the other hand, needs to be investigated. The links between observable behaviour and existing theoretical models of higher-level cognition are less well established, and results gained with comparatively simple problems in laboratory experiments may not be generalisable to user interaction with complex systems.

(7) HCI phenomena which have not been sufficiently well described and understood should not be expressed as formalisms and investigated by methods of scientific experimentation (Green et al., 1988); instead, exploratory research is required to obtain precise descriptions as a basis for further research.

(8) Providing system designers with rapid feedback and suggestions for improvement in accessible language is a worthwhile activity which may improve the usability a specific system. To make a contribution to HCI research, however, such observations have to yield sufficiently detailed descriptions of user-system interaction.

(9) HCI research should use both qualitative and quantitative methods (Newell & Card, 1986). The semantic prejudice which associates experiments and quantitative data with good science and clear, valid and applicable results must be overcome. Anecdotal science and woolly arguments are not avoided by conducting experiments. Experiments which are inappropriate, or badly planned or controlled, can set back research. Anecdotal science is avoided by studies which are well planned and executed, and meticulously documented for the inspection by other researchers.

(10) In order to progress from descriptions, these have to form the first step of a theory-building process. Theories provide the most systematic way of building, synthesising and integrating scientific knowledge (Strauss & Corbin, 1990). Thus, observations and results of specific evaluations must be used as a basis for the formulation of HCI theories (Newell & Card, 1985).

(11) The need to formulate HCI theories does not, however, imply that these must be engineering-style theories. The debate in this chapter has identified good arguments that there are HCI phenomena which cannot be expressed in this fashion. Having rejected the proposal of artifacts as HCI theories, other approaches to constructing non-mathematical theories from observations have to be considered. Such approaches have been devised in social sciences. Grounded theory (Glaser & Strauss, 1967), for instance, offers a framework for deriving theories from observations which exactly fits HCI's requirements, without imposing the constraints of the traditional science approach[8]. Grounded theory takes

[8] According to Strauss & Corbin (1990), the analytic procedures of grounded theory are designed to:

the route of data-analysis-theory instead of the theory-hypothesis-test path pursued in traditional science. Theories constructed in this manner can not only be used to explain observations, but also provide a framework for deriving subsequent action, which suits the applied nature of HCI.

As a small diversion, it is intriguing to consider that grounded theory may not only be used to construct theories for specific HCI phenomena; it may also provide a framework to evolve a conception for the discipline – which is, after all, a meta-theory. Dowell & Long (1989) suggest that principles can be derived from existing craft knowledge by induction, and from existing scientific knowledge by deduction; grounded theory mixes the two methods and uses qualitative and quantitative data. Given that not all HCI researchers agree with Dowell & Long's conception, the HCI research community should consider treating HCI knowledge as data from which the conception for the discipline can be generated by using grounded theory methods.

To return to the pursuit of a specific research undertaking in HCI, it has to start by establishing existing knowledge of the phenomenon under investigation both in HCI and related disciplines. The research method and tools will depend on the nature of the phenomenon under investigation, and state of existing knowledge.

The research topic of this thesis - i.e. the phenomenon under investigation - is the mental model which the user forms of a computer system. The appropriateness of this model is thought to determine the success or failure of the user's interaction with a system. The way in which information about the system is presented to the user is supposed to influence the model which the user builds. If this is true, knowledge about which models help users to interact successfully with computers, and how to present information to users such that they form these models, could

---

"1. Build rather than only test theory.

2. Give the research process the rigor necessary to make the theory "good science".

3. Help the researcher to break through the biases and assumptions brought to, and that can develop during, the research process.

4. Provide the grounding, build the density, and develop the sensitivity and integration needed to generate a rich, tightly woven, explanatory theory that closely approximates the reality it presents. " p. 57

make a very important contribution to system design. Newell & Card (1985) identified mental models as a candidate for an engineering-style theory. In order to decide whether this is an appropriate form and methods for this phenomenon, existing general knowledge on mental models needs to be examined; this is done in Chapter 3. Existing HCI theories and data on mental models are reviewed in Chapter 4.

3 Theories of Mental Representation and Mental Models

Theories of mental representations in general, and mental models in particular, deal with form and function of individual knowledge. The central question is how human beings represent information mentally, and how they use that information to interact with the world in adaptive ways. Users' knowledge about computer systems are a specific type of mental representations. Mental representations have been investigated by researchers in philosophy, cognitive psychology and – more recently – cognitive science. The aim of this chapter is to examine existing general knowledge from those disciplines which may contribute to the understanding of the specific HCI problem: the representations users form of computer systems they are interacting with, and how these representations influence their interaction with the system. Three general theoretical approaches are examined: Paivio's (1986) classification of mental representations, Johnson-Laird's (1983) theory of mental models, and a collection of work on mental models of natural phenomena and devices edited by Genter & Stevens (1983).

3.1 Physical and mental representations

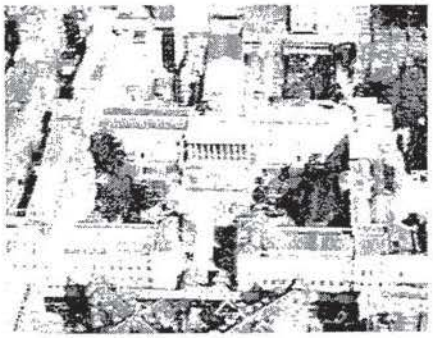
physical representations	picture-like	language-like
examples	photographs drawings maps diagrams	human-language formal systems: maths, symbolic logic computer programs
properties	analogue iconic continuous	non-analogue non-iconic digital/discrete
mappings	referentially isomorphic	referentially arbitrary

Table 3: Types of physical representations (after Paivio, 1986)

Historically, mental representations have been interpreted by analogy with physical representations, i.e. descriptions and classifications devised for physical

representations have been applied to mental representations (Paivio, 1986). Physical representations can be picture-like or language-like (see Table 3).

Picture-like representations, such as photographs and maps, are *analogue, iconic* and *continuous*. *Language-like* representations, which include natural language and mathematics, are *non-analogue, non-iconic,* and *digital* or *discrete*. The crucial distinction between picture-like and language-like representations lies in the degree of arbitrariness of the mapping relation between the representation and the represented phenomenon: picture-like representations map onto objects in a non-arbitrary manner, i.e. they are *referentially isomorphic,* whereas with language-like representations, the mapping is constructed in a *referentially arbitrary* fashion. It is important to note that following this definition, many representations referred to as "pictures" in everyday language are not picture-like, since the mapping relationship between a picture and what it represents can be arbitrary. Picture-like and language-like should be seen as two ends of a continuum, rather than discrete categories: many representations combine picture-like and language-like properties (see Figure 1).



Picture-like ⟷ Language-like

Figure 1: Picture-like and language-like representations

In everyday language, the distinction between picture-like and language-like representations is often referred to in terms of concrete *vs.* abstract representations. *Concrete* and *abstract* are, however, the poles of a related but separate a dimension which applies to both picture-like and language-like representations. Typically, a photograph is more concrete than a drawing of the same object, and in natural

language we distinguish between concrete and abstract nouns. Finally, there is the holistic *vs.* componential dimension. A picture-like representation is typically seen as *holistic*, but can be composed from components (e.g. identikit pictures). Language-like representations are typically *componential* (e.g. human language is composed of phonemes), but may use holistic visual symbols (e.g. numbers).

The properties of picture-like and language-like representations have implications for their use. A picture-like representation preserves all distinctive features or characteristics of the object it represents; the mapping between the object's features and their corresponding features in the representation is done in a non-arbitrary fashion. Due to the replication of distinguishing features and the relationships between them, a picture-like representation usually exhibits a great degree of similarity with the physical object which it represents: it is recognised as a representation of the object. For example, a person who is familiar with the Eiffel Tower will recognise a picture-like representation (e.g. a photograph or drawing) of the Eiffel Tower as such. Similarly, a person who has seen a picture-like representation of the Eiffel Tower in a guidebook is likely to recognise the Eiffel Tower when they set sight on it. In contrast, a person who has only come across a language-like representation of the Eiffel Tower (the words "Eiffel Tower") may not recognise the spectacular iron construction in front of them as the Eiffel Tower[9]. The mapping from an object to its language-like representation can be entirely arbitrary, or follow one of an infinite number of mapping rules. In any case, the connection between an object and its language-like representation, or the mapping principle which connects them, has to be established in a person's mind for the language-like representation to be recognised as a representation of what it stands for. In other words, the representation will only have its intended effect if the connection or the mapping rules have been learnt previously by the viewer.

---

[9] This statement applies to the simple language-like representation "Eiffel Tower." A detailed verbal description of the Eiffel Tower could, of course, relay all its characteristic features to such effect that a reader would recognise it. At this point we enter the realm of *"Why a diagram is (sometimes) worth ten thousand words"* (Larkin & Simon, 1987), which goes beyond the basic classification of physical representations and their properties discussed here.

Picture-like representations may thus seem to be the obvious choice for representations which have to be recognised by a wide range of people, who we cannot be sure know the connection with the represented object or mapping rules applied to derive the language-like representation. Yet, the very properties which facilitate recognition of picture-like representations of known objects – the preservation of features and their relationships – limit their *reductive power.* A picture-like representation may represent an object on a smaller scale, such as an architect's model of a building; this process of reduction may not reproduce some of the more detailed features. Another process is to collapse the object, such as in a photograph; this may not reproduce all dimensions of the object. If we aim to represent a large and complex object through a picture-like representation, the representation will achieve some reduction, but still be fairly large and complex – *viz.* an architectural drawing of a large and complex building. For this reason, most cultures have, as the information to be communicated increased in complexity, moved from systems located on the picture-like end of the continuum towards the language-like ones (Kuckenberg, 1989). Language-like representations do not have to preserve distinctive features and their relationships. Moreover, they can be used to represent more than just physical objects – for instance, abstract notions such as concepts and ideas. Information can be aggregated and combined by employing rules or principles. Elements in language-like systems can represent more than one object or meaning – i.e. they can be *polysemic*; picture-like representations tend to be *monosemic, i.e.* represent one object only (Goodman, 1984). By employing these mechanisms, language-like representations can achieve a high degree of reductive power: an accumulation of things, ideas, concepts or events can be expressed in a very compact representation. To identify the intended meaning of such a representation, people interpreting it have to be familiar with the meaning of the representation or rules for associating it with an object; they may also have to identify other factors which impact meaning, such as the context in which the representation is used. If the connections or rules are not – or only partially – known by a person interpreting a language-like representation, they may not be able to work out its meaning, or they may interpret it in a different manner from the intended one.

Many representations try to exploit the advantages of picture-like representations whilst trying to achieve a high reductive power at the same time. In reality, distinguishing between a full isomorphic mapping and a partial one is difficult, since it can be a point of argument which features of an object are characteristic and therefore have to be preserved. Other representations combine picture-like and language-like elements to construct a representation which is as compact as possible, but retains one or more characteristic features to direct interpretation towards the intended meaning.

The distinction between picture-like and language-like representations and their properties is an important one. The brief discussion in this section has examined the properties of these different types of physical representations, and highlighted potential consequences of their application (reductive power, existing knowledge and mental workload). The dimensions and properties have been established for physical – externalised – representations; whether the historical assumption that mental representations are analogous to these is a different question.

3.2 Mental models

One of the most influential theories to be formulated in cognitive psychology in recent years is Johnson-Laird's (1983) theory of mental models. The theory seeks to provide a general explanation of human thought; at its core is the assertion that humans represent the world they are interacting with through mental models. Johnson-Laird credits Craik (1943) with the original statement of this idea. In order to understand a real-world phenomenon, a person has to hold a what Johnson-Laird (1983) describes as a *working model* of the phenomenon in his or her mind. Mental models are not imitations of real-world phenomena, they are simpler. They do not correspond completely to what they model – Johnson-Laird argues that adding information beyond a certain level does not increase its usefulness. A mental model which explains all aspects of the phenomenon that a person interacts with is an appropriate one. In order to provide explanation, it has to have a similar structure to the phenomenon it represents; it is this similarity in structure which enables the holder of the model to make mental inferences about the phenomenon which hold true in the real world. Since the choice of structure is not arbitrary, but analogous to

that of the phenomenon, the mapping relationship between mental models and the phenomena they represent is a referentially isomorphic one. Thus, mental models do exhibit similar characteristics to a picture-like representation; and, as with a picture-like representation, whether a model is appropriate or not can be a point of argument.

The theory of mental models demolished an assumption which was until then prevalent in psychological theories of reasoning: that humans employ a kind of mental logic, which is similar to the propositional logic employed by logicians, when making inferences about the world. Johnson-Laird does not argue that human beings are incapable of logical inference, nor does he seek devalue the concepts of mathematical logic. He simply seeks to explain the puzzle, posed by empirical and observational facts, that human beings are capable of logical inference, yet often deliver answers and decisions that cannot be explained in terms of logic reasoning. A series of empirical investigations using the Wason-task (Wason & Johnson-Laird, 1972) illustrates this: only 12% of subjects were able to solve a simple logical problem presented to them in abstract form; yet, 60% were capable of solving a formally equivalent version of the same problem which uses objects and a context which they are familiar with. The results of these experiments – which have been replicated many times with different subject groups in different countries – demonstrate that most people's reasoning is primarily influenced by the *content-relatedness* and *form* of the information presented. People are more likely to solve a problem correctly when they have relevant background knowledge which can be employed. They are more likely to relate relevant existing knowledge to the problem presented if the structure of the information presented is compatible with the structure of existing knowledge. Relevance and compatibility are defined in terms of meaning (i.e. semantic content) rather than logic (i.e. syntactic structure).

Johnson-Laird (1983) proposes that reasoning about a problem is facilitated if a person utilises a mental model that represents the relevant information in an appropriate fashion for the problem to be solved. This raises the question of how mental models are constructed and applied.

Johnson-Laird (1983) proposed that there are three types of mental representations:

(1) *propositional representations*, which are pieces of information resembling natural language;

(2) *mental models*, which are structural analogies of the world; and

(3) *mental imagery*, which are perceptual correlates of models from a particular point of view.

Information is represented at several different levels of abstraction. Indeed, the form of representation can vary from one level to the next: propositional representations are referentially arbitrary, whereas the mental models are analogues, and therefore referentially isomorphic.

General procedures	1 A procedure that begins the construction of a new mental model, whenever a proposition makes no reference (implicitly or explicitly) to information held in the current model of the world.
	2 A procedure that adds entities, properties, or relations to a current model, if at least one entity referred to in this proposition is represented in the current model.
	3 A procedure that integrates two or more existing models, whenever a proposition interrelates entities in them.
	4 A procedure that verifies that all asserted properties or relations hold in a new model, whenever all entities referred to in a proposition are represented in a current model.
	5 A procedure that adds a property or relation to a current model.
Recursive procedures	1 If a proposition is true in a current model, then checks are made to see whether there is an alternative model of the previous propositions that is inconsistent with the current propositions. If there is such a model, then it is assumed that the current proposition is only possibly true. If there is no such model, it is assumed that the proposition is true.
	2 If a proposition is false in a current model, then either the model of the previous propositions is modified to bring it in line with the current proposition, or the current proposition is interpreted as being inconsistent.

Table 4: Procedures for mapping propositional representations into mental models (Johnson-Laird, 1983)

Propositional representations form the building blocks from which a mental model is constructed. They have been described as the *mental echo* of the information available in the real world (Manktelow & Jones, 1987), encoded in a verbal format. Thus, a person holding a propositional representation will be able to recall the information contained in the propositional representation verbatim. Since propositional representations are neither integrated nor elaborated with other information held in memory, information encoded solely in terms of propositional representations is (a) difficult to remember; and (b) does not enable a person to perform operations like generalisation or inference on the information encoded in them. These properties of propositional representations are closely linked to the mechanisms which govern the working of short-term memory.

The process by which propositional representations are mapped into mental models is called procedural semantics. Johnson-Laird (1983) states that all mappings have to be described at the level of procedures. The procedures specify how a given mapping can be carried out, in a way that can be followed by a simple machine[10]. Johnson-Laird (1983) specifies five general and two recursive procedures for mapping propositional representations into mental models (see Table 4). For each new item of incoming information, a search is made to ensure that the proposition is consistent with any earlier information encountered. If an appropriate model can be found to accommodate the proposition, the model will be cued, applied, and perhaps modified. If no appropriate model can be found, the relevant procedures will be employed to construct a mental model from scratch (construction *de novo*). It is worth emphasising again that – as the name procedural *semantics* indicates –

---

[10] For Johnson-Laird (1983), any theory of the mind has to be computational, since he subscribes to functionalism (see Chapter 1 in Paivio (1986) for an account of the functionalism *vs.* empiricism debate). The following quote summarises his position:

"... it follows not only that scientific theories of mentality can be simulated by computer programs, but also that in principle mentality can be embodied within an appropriately programmed computer: computers can think because thinking is a computational process. [...] ... any future theory of the mind will be completely expressible within computational terms." (p. 10)

It is outside the scope of this thesis to engage in a discussion of this position, and whether the computer is indeed the ultimate metaphor of the mind. The author does not subscribe to functionalism, but this has no immediate implications for the topic of the thesis.

evaluation of whether a proposition is true or false is conducted in terms of *meaning* attached to the proposition, not syntactic rules of truth tables used in logic.

3.3 Mental models of natural phenomena and devices

1983 was the year which put mental models on the cognitive research map: it saw not only the publication of Johnson-Laird's (1983) seminal volume, but also the publication of a collection of papers with the same title edited by Gentner & Stevens (1983). Despite having the same title, the two publications represent very different directions in research on mental representations.

The collection surveys and discusses a variety of internalised and externalised models, ranging from "naive" models of natural phenomena, such as electricity, to instructional models of scientific concepts, often used for teaching purposes. The collection also contains two of the earliest papers of mental models of computer systems by Norman (1983) and Young (1983), which are reviewed in detail under 4.2.1 and 4.2.3, respectively. Unlike Johnson-Laird (1983), who seeks to formulate a cognitive theory to explain human thought in general, the authors of each of these papers seek to model beliefs about a specific domain – i.e. a natural phenomenon or device. The common theme which emerges from the study of diverse domains is analogical reasoning: people attempt to form an understanding of an unknown phenomenon by transferring inferences from an existing mental model to the new phenomenon. Collins & Gentner (1987) suggest that mental models are always constructed through analogical reasoning; on the other hand, the process of structure-mapping between two domains seems to require that the two phenomena have similar surface characteristics (Genter, 1983).

Johnson-Laird (1983) describes mental models as structural analogues, but this is different from analogical reasoning as discussed in Genter & Stevens (1983). Johnson-Laird focuses on the structural analogy between the externalised phenomenon and its internalised representation; Genter & Stevens focus on the transfer from one internalised model to another. For them, *structural similarity* includes surface characteristics and syntactic features, whereas Johnson-Laird focuses on meaning. As Payne (1992) points out, Johnson-Laird's theory encompasses the basic theoretical commitment shared by all authors in the Genter &

Stevens volume: that people's existing knowledge has a considerable influence on their reasoning about a new problem, phenomenon, device or idea. Johnson-Laird's theory of mental models is more developed in that it specifies the format of representations, and procedures which are used to operate them.

3.4 Conclusions

The aim of this chapter was to examine existing general knowledge on mental representation and mental models, and determine if any of the existing knowledge, or part of it, can be applied to the specific HCI research undertaking of this thesis – users' mental models of computer systems and their influence on users' interaction with the system. The examination of three contributions on mental representation and mental models has led to the following conclusions:

(1) It is worthwhile to consider physical representations and their properties, since computer systems employ physical representations of:

- themselves, or parts of themselves (e.g. functions)
- objects
- tasks

with which users interact.

(2) Designers can employ picture-like or language-like representations for users to interact with, or representations which lie somewhere in between on the continuum. Picture-like representations may be easier to recognise as representations of the functions, objects and tasks they represent; on the other hand, they have limited reductive power, thus, technical constraints of computer displays (limited screen size, resolution) may it difficult to incorporate representations which are at the picture-like end of the continuum. The first generation of professional computer users (see 2.1) interacted with computer systems using language-like representations (e.g. command languages). Language-like representations are very powerful, but require that the user knows what they represent, or knows the mapping rules required to derive the meaning. This requires learning on the part of the user, and represents a cognitive overhead when using the system. Many designers employ

representations which combine picture-like and language-like elements, such as pictograms and computer icons. They are more compact than pure picture-like representations, and retain at least one characteristic feature of the what they represent. This may guide the user's interpretation, but the connection between them and the object, function or task represented still has to be learnt.

(3) According to Johnson-Laird's (1983) theory of mental models, people construct mental models (which have picture-like properties) from propositional representations (which have language-like properties). Following this theory, users' ability to interact with a computer system depends on whether they have an appropriate mental model of the system. An appropriate model does not have to be complete and correct in every detail; all it has to do is explain the functions and behaviour of the system which are relevant to the user. The cueing and construction of such a model can be supported by the way in which information is presented to a user: form and content of information determine which existing models are cued, and how new information is mapped onto existing or new models through the mechanism of procedural semantics. Applied to HCI, this means that designers would have to identify suitable existing knowledge to be cued, and present relevant information about the system in the context and form which directs the model-building process towards the intended mental model.

(4) The elaboration under (3) demonstrates how Johnson-Laird's (1983) theory of mental models might be applied to the design of user interfaces which support the construction of appropriate users' mental models of itself. According to Johnson-Laird (1983), the theory provides a general explanation of human thought; however, its empirical support stems from experiments involving short (mainly syllogistic) reasoning tasks. The application of the theory to mental representations of computer systems remains to be demonstrated. There are three interconnected arguments as to why the theory may not be applicable to users' interaction with computer systems:

- The collection by Genter & Stevens (1983) provides some evidence that mental models and the mechanisms by which they are constructed may differ according to the task or problem domain.
- The task of interacting with a computer system has very different cognitive parameters (amount of knowledge, duration and frequency of interaction) from those of a short reasoning task.
- As Payne (1992) points out, Johnson-Laird's theory is essentially a mental models theory of interaction with a particular artifact, namely text. The specific HCI problem involves interaction with an artifact – a computer system – which has at least one additional dimension – "write" – in addition to the "read" dimension involved in interacting with text.

(5) Users may employ analogical reasoning – using an existing model from a different domain – to explain the workings of a computer system. HCI literature provides examples of users as well as system designers and instructors attempt to use this mechanisms (e.g. *"a word processor is like a typewriter"* – Carroll & Mack, 1984). Analogical reasoning requires an existing mental model which has sufficient similarity to act as the source of the mapping; this raises the question whether users are likely to have existing models which are similar enough to explain the behaviour of computer systems.

4 Mental models in HCI: a review of the literature

The purpose of this chapter is to establish the state of existing knowledge about mental models in HCI by providing a summary and critical review of published theoretical and empirical research work on this topic to date. The result of this review provides the starting point for research required to advance the discipline's knowledge on this topic.

Given the interdisciplinary nature of the HCI, and the comparatively short history of this particular research topic, it is not surprising to find a certain diversity in the terminology used in the published research. The differences in concept and nomenclature are outlined in 4.1; this section also explains the choice of terms employed to describe the work presented in the remainder of the thesis.

Section 4.2 describes and compares existing theories and hypotheses of mental models in HCI. The comparison concludes that all theories of mental models in HCI are – explicitly or implicitly – based on a core set of assumptions:

(1) Users form a mental model of the internal workings of the computer systems they are interacting with.

(2) The content and structure of that mental model influences how users interact with a system.

(3) The content and structure of a mental model can be influenced by selecting what information about the system is presented to users, and how it is presented.

(4) More detailed knowledge of how users construct, invoke, and adapt mental models could be used to provide guidance for user interfaces and user training which help users to form appropriate models, and therefore make an important contribution to HCI.

The theories do, however, not agree as to the details of the following:

(1) the content and structure of the mental models held by users;

(2) how exactly users' mental models influence their interaction with computer systems; and

(3) the nature of the process through which mental models are constructed.

In 4.3, the empirical evidence on mental models in HCI generated to date is considered. The review concludes that results are fragmented and do not provide an endorsement of any particular theory, or conclusive evidence to decide any of the points on which opinions diverge. This is – at least in part – due to the disparate set of methods and measurements employed in the empirical studies.

The conclusions from the review, and resulting aims for the research presented in the remaining chapters of the thesis, are presented in 4.4. The main conclusion is that, since existing knowledge is fragmented and patchy, basic research needs have to be addressed to further knowledge on mental models in HCI.

4.1 Terminology

The term *mental model,* which in the works of Johnson-Laird (1983) and Gentner & Stevens (1983) refers to internalised[11] representations of a device or idea held in the mind of one or more persons, has been applied to quite a diverse assembly of representations in HCI. With the various theories of mental models, a whole range of terms has been introduced. The lack of a unified terminology is confusing, especially when different authors use different terms to describe the same type of model, and the same term is sometimes used for very different types of models (Nielsen, 1990). This subsection provides an overview of the terms used in the context of the theories with which they are connected (see Figure 2).

The first attempts at devising a terminology for a theory of mental models for HCI can were those by Norman (1983) and Young (1983).

4.1.1 Models of users and systems

Norman (1983) introduced the distinction between four different types of representation which influence user-system interaction: the computer system with which the user is interacting is the *target system.* The user constructs a *user's mental model* of that target system. The target system should communicate the *conceptual*

[11] The distinction between internalised and externalised models is introduced as part of Nielsen's (1990) taxonomy in 4.1.4.

model, which is an accurate, consistent, and complete representation of the target system held by the designer, or an expert user, of the system. The user's mental model is formulated through interaction with the target system, and constantly modified throughout the interaction. The *scientist's conceptualisation* is a model describing the content and structure of the user's mental model.

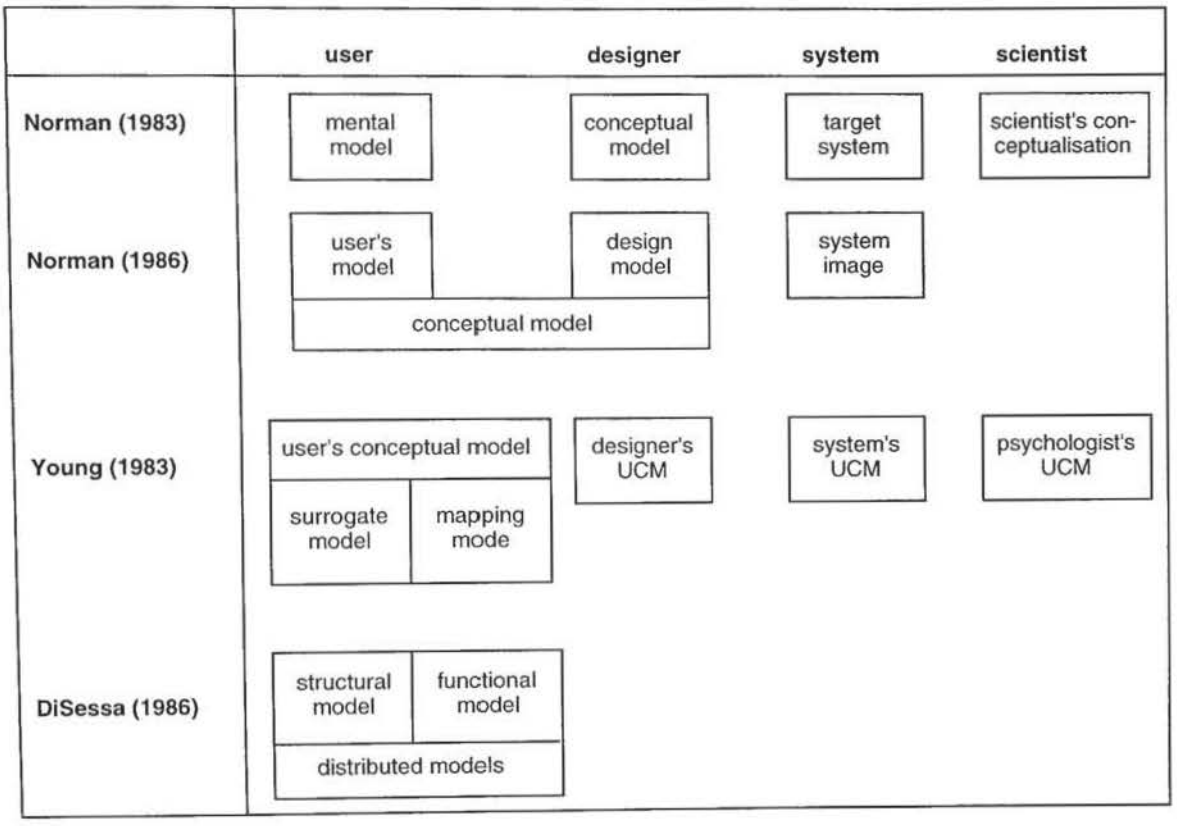


Figure 2: Overview of terminology

Norman revised these terms shortly afterwards. In 1986, he presented a more refined theory of the various models and how they interact, as part of the *cognitive engineering* framework introduced in the seminal volume on *User Centred System Design*, edited by himself and Stephen Draper (Norman & Draper, 1986). The notion of the target system has been extended to that of a physical model of the system which is available to the user – apart from the screen display, the *system image* includes input/output devices, any documentation, training, error and help facilities. The designer constructs a *design model* which is communicated through the system image. The user develops a *user's model* through interaction with that system image. The term *user's model* encompasses both the notion of the idiosyncratic model

developed by an individual user, and the idealised model which the designer considers an appropriate one for users to develop. (The latter, therefore, is the *designer's conceptualisation of the user's model*, akin to the scientist's conceptualisation of the user's mental model in the previous version.)

In line with the model of user-system interaction put forward in the cognitive engineering framework, the relationship between the two internalised models is a circular one, rather than the one-way mapping described in Norman (1983). The choice of design model is linked to the idealised user's model, which itself is chosen to tie in with users' background knowledge and experience. Together, design model and user's model form the conceptual model of the system.

4.1.2 Surrogates and mappings

In the same volume as Norman (1983), Young (1983) introduced a distinction between different types of representations held by users. He defines the term *user's conceptual model (UCM)* as:

> *"... a more or less definite representation or metaphor that a user adopts to guide his actions and help him interpret the device's behaviour." (p. 35)*

From this definition, we can assume that the UCM is equivalent to the *user's mental model* or *user's model* in Norman's (1983, 1986) terminology. Young (1983) also has externalised versions of that model, the *Psychologist's UCM* (similar to the scientist's conceptualisation), and *Designer's UCM* (similar to the conceptual model)[12]. This choice of terms may imply that the user's internalised representation is the basis from which research and design models should be derived.

Furthermore, Young (1983) introduces a distinction between different types of internalised representations. *Surrogate models* are mechanistic, if highly simplified, accounts of a device (e.g. computer system), which can take the form of a scale model, or be written using a formal or informal notation. The choice of the term *surrogate* implies that such a model can be used in place of the system itself. This

---

[12] Like Norman (1986), Young distinguishes these from *user models*, such as those employed in Intelligent Tutoring Systems, which are a conceptualisation of certain characteristics of a user, embodied in the system itself.

may seem a desirable model for users to hold. The drawback is that for a complex system, it will take considerable time and effort to acquire such a model, and continuous use to maintain it. Surrogate models are, therefore, the type of model we would only expect to find in expert users of computer systems.

With the second type of model – *task-action mapping models* – Young linked the internalised representation of the system to the real-world task which users have to perform. Moran (1983) had pointed out that users of computer systems have to translate the tasks which they have to perform in the real world (e.g. deleting a paragraph from a document) into the task which they have to perform on the computer (e.g. inserting a beginning and end marker to mark the paragraph, then select the delete function). Task-action mappings describe the structure of a real-world task and the actions needed to perform that task, and provide a direct mapping between the task and the corresponding actions. A set of central *core tasks* is chosen, and a corresponding set of *core action sequences* designed. The links between core entities (tasks and actions) form the core of the mapping model. Other tasks can then be expressed as variants of the core tasks, and be mapped onto the appropriate actions. The core of the mapping acts as a communication channel, carrying all the mappings between the two domains. This requires the core entities to have a corresponding *internal structure*, i.e. the mapping between core tasks and core action sequences has to be isomorphic. Young (1981) uses the example of an algebraic calculator for a mapping model: the operations that have to be performed can be mapped onto doing the same task with paper and pencil. A task-action mapping model would allow competent use of a system for a particular task, even though the user has no detailed knowledge of the system and how it works. We can find examples of such mappings at work in everyday life: competent drivers who do not know how an internal combustion engine works, cooks who produce delicious meals without knowledge of the physics and chemistry involved in the processes, and millions of telephone users.

4.1.3 Distributed models

The distinction between knowledge of the internal structure of a system, as opposed to knowing how to use a selected range of functionality to perform real-world tasks,

has been developed further by DiSessa (1986). *Structural models* (similar to surrogate models) provide users with a detailed understanding of the system which is defining and independent of a specific task, whereas *functional models* (similar to task-action mappings) represent selected properties of the system required to perform a specific real-world task. DiSessa (1986) discusses the implications of acquiring and applying the two types of models (see Table 5). Functional models may seem to be ideal models for non-expert users who want to use the computer as a tool, since they take less time and mental effort to acquire and maintain. Alas, the answer is not that simple. Users may want to use the computer as a tool for their job. But unlike traditional tools, most computer systems are not dedicated to one specific task: most computer systems are many-tools-in-one, employed for a range of tasks. Dedicated *walk-up-and-use* systems, such as cash dispensers and ticket vending machines, are the exception. Therefore, users would have to acquire and manage a multiplicity of overlapping functional models; this raises the question at which point the mental effort required to do this exceeds the effort associated with acquiring a structural model. In addition, experience has shown that it is unlikely that all task-function mappings will be known at the design stage: users often apply systems to tasks which neither they nor the designer had envisaged at the design stage.

	Structural model	**Functional model**
Size of model	complete	reduced
Acquisition		
time	long	short
mental effort	high	low
Application		
mental effort	high	low
scope	general	restricted

Table 5: Properties of structural and functional models (DiSessa, 1986)

DiSessa (1986) suggests that a neat division between structure and function, and corresponding users' models, is not possible. The multiplicity of functions performed by most systems is one reason. The other is that most users do not learn

and reason about systems – or other devices – in an orderly fashion, and do not develop neat and tidy models. This statement confirms Norman's (1983) observations regarding characteristics of users' models:

(1) Users' models are *incomplete.*

(2) Users' ability to "run" models is *severely limited.*

(3) Users' models are *unstable*: users tend to forget detailed features of the systems they use.

(4) Users' models *do not have firm boundaries,* and therefore similar devices and operations are often confused.

(5) Users' models are *"unscientific"*: users tend to stick with certain ways of doing things, even when they know better ones.

(6) Users' models are *parsimonious*: users tend to trade off extra physical actions for reduced mental complexity.

DiSessa (1986) introduces the term *distributed models* to describe the actual state of internalised models, :

> *"... a patchwork collection of pre-existing ideas in the learner, "corrupted" to new ends." (p. 209)*

Distributed models are, therefore, not neat structural or functional models. Rather, they are accumulations of multiple partial explanations users hold of a system, tied in with their existing knowledge and experience.

Nielsen (1990) also uses the terms *distributed models* and *structural models,* but these have somewhat different meanings (see 4.1.4).

4.1.4 A taxonomy of mental models

In an attempt to bring some order into the great model confusion of HCI, Nielsen (1990) proposed a meta-model to classify models of user-system interaction. This meta-model encompasses a wider range of models than those which are the topic of this thesis; this subsection extracts the dimensions and part of the taxonomy which are relevant to mental models.

The 7 elements or participants of models are:

- **U** – the user;
- **D** – the designer;
- **C** – the computer system;
- **M** – manuals and other documentation of C;
- **T** – the task performed by the user[13];
- **W** – the surrounding world in which U performs T[14] ;
- **R** – the researcher looking at any of the above.

These elements can be combined, using a simple notation, to denote who holds a representation of what. For example, a user's model of a computer system is UC, the designer's model of a user's model is D(UC), and so on. This notation offers a parsimonious way of distinguishing between different models.

An additional level of description and analysis can be added by placing a model on Nielsen's (1990) 6 dimensions.

(1) Internalised *vs.* externalised models

A fundamental distinction, which has already been applied in the analysis of terminology in the previous section, is that between *internalised* and *externalised* models. *Internalised* models reside inside the human mind (e.g. the user's model of the system, UC) or the computer system (e.g. the computer's model of itself, CC). *Externalised* models are represented in some explicit form in a medium outside the human mind or the system (e.g. a designer's documented conceptualisation of the computer system, DC). The representation can be in form of a natural or formal language description, or a diagram or drawing.

---

[13] Nielsen (1990) mentions tasks and the surrounding world, but does not expand on their models and relationship to other models; the analysis of theoretical contributions in 4.2, however, shows that they do play an important role.

[14] See above.

(2) Structural *vs.* distributed models

This dimension distinguishes between the partial, overlapping, somewhat messy collection of *distributed models* described in 4.1.3, and the neatly constructed *structural models*, which follow a guiding principle[15]. Even though Nielsen (1990) does not state this explicitly, we can expect most internalised models (such as UCs) to be distributed. Most *externalised* models, including those used for documentation and training purposes (MCs), would be expected towards the *structural* end of this dimension.

(3) Generic *vs.* instantiated models

This dimension offers a way of distinguishing the two types of model subsumed in Norman's (1986) *user's model* (see 4.1.1). The idiosyncratic model of an individual user (UC) is an *instantiated* version of the *generic* user's model envisaged by the designer (D(UC)) or researcher (R(UC)). All generic models are users' models – how well an individual user's model fits into the generic descriptions is an open question at this point.

(4) General *vs.* specific models

This dimension is related to generic *vs.* instantiated, but distinguishes the level of abstraction at which a model is described. *General models* are aimed at describing users and computer systems in general terms, whereas *specific models* would be subsets representing a reasoning mechanism employed by a particular user group or describe the design model of a particular application.

(5) Descriptive *vs.* analytic models

This dimension expresses the different levels of formalisation and abstraction at which a model is represented. Natural language descriptions of models such as UCs or MCs are just that – descriptive,

---

[15] There difference between structural models as defined by Nielsen (1990) and DiSessa (1986) is therefore the general *"organised along some guiding principle"* rather than the more specific *"mental analogue of the computer system."*

whereas models represented using a formal method which affords further analysis (e.g. checking for consistency), are analytic. The extent to which analytic models have been employed in the design process to date varies. Analytic models of users and systems have proved to be more popular with researchers (RCs and RUs[16]) than with designers, who mainly use a mixture of informal diagrams and natural language for D(UC)s and DCs (Newman & Lamming, 1995). HCI methods based on task analysis (e.g. Diaper, 1989), on the other hand, seem to have made a more successful transition from R(UT) to D(UT).

(6) Static *vs.* dynamic models

This dimension distinguishes between models which do not change continuously over time *(static models)* and those which change continuously *(dynamic models)*. There is a strong indication that most internalised models, especially UCs, are dynamic. Norman (1983) stated that user's models are constantly refined during user-system interaction (see 4.1.1.); he also observed that details of models are easily forgotten (see 4.1.3). Externalised models, such as DCs, may be dynamic during the design process, but do become static after the design has been completed.

The notation and the dimensions will be employed to clarify the exact nature of models involved in the theoretical and empirical work reviewed in this section.

4.1.5 Terms used in this thesis

Given the problems with the diverse and ambiguous terminology demonstrated in this section, a case could be made for avoiding existing labels altogether, and discuss the various models solely in terms of Nielsen's (1990) notation. We know, however – not least from Johnson-Laird's (1983) work – that the use of an unfamiliar abstract notation is likely to increase the cognitive load and adversely affect the reasoning ability of a reader. By way of compromise between an unambiguous but unfamiliar

---

[16] An example of an analytic RC is Payne & Green's (1986) Task-Action Grammar (TAG); and of an RU the Human Processor Model (Card et al., 1986).

notation, and familiar but ambiguous labels, commonly used terms will be employed in this thesis, with Nielsen's (1990) labels and dimensions added.

Most of the published work on mental models in HCI makes reference to Norman (1986); his theory of conceptual design (as outlined 4.2.1) has been the most influential proposal as to how users' models can be harnessed to design more usable systems. The two fundamental models are *users' models* and *design models*[17]. Table 6 describes the models more closely by indicating their likely position on each of Nielsen's (1990) six dimensions.

Dimensions	User's model(UC)	Design model (DC)
internalised – externalised	internalised	externalised
structural – distributed	distributed	structural
generic – instantiated	instantiated	generic
general – specific	specific	specific
descriptive – analytic	descriptive	descriptive
static – dynamic	dynamic	static

Table 6: User's models and conceptual models

The entries should be seen as an illustration of how a specific model could be described in detail using the dimensions, rather than assigning these dimensions to all user's models and design models. The entries should be read as *"probably towards the distributed end of the continuum"*. To give an example, we expect most users' models to be distributed, but some expert users' models may be towards the structural end.

To avoid confusion between the structural-distributed continuum and the structural and functional models described by DiSessa (1986), the terms *surrogate model* and *task-action mapping* (Young, 1983) will be applied to distinguish the two different types of users' models identified by these authors. Some additional terms will be

[17] For some reason, the term *conceptual model* (Norman, 1983), has persisted after the introduction of the term *design model* (Norman, 1986). It is more widely used, but with a wide variety of meanings. In the interest of consistency and unambiguity, the term *design model* has been adopted together with the other labels from the revised (1986) version.

defined the following section, in the context of the theoretical contributions in which they are introduced.

4.2 Theoretical contributions

This subsection introduces and analyses 5 theoretical contributions on user's models in HCI. The purpose of the analysis is:

(1) to identify the types of models involved in the theories;

(2) to establish how these models are seen to influence users' interaction with computer systems; and

(3) to outline how the theories could be applied to designing user interfaces which facilitate learning and use of computer systems.

In order to determine similarities and differences between the theoretical contributions, the models and mappings introduced in each contribution will be described using Nielsen's (1990) notation and dimensions, as introduced in 4.1.4. A list of all models identified in this section is provided in Appendix 1.

4.2.1 Conceptual design

Most of the theoretical and empirical contributions reviewed in this chapter assume that a well-designed system and user interface will allow the user to develop an appropriate internalised model of that system, which in turn facilitates users' learning of, and interaction with, the system. This assumption is at the core of the *conceptual design* approach pioneered by Norman (1983; 1986). He postulated that if the designer gets the design model (DC) "right", and communicates this model successfully through the system image, users interacting with the system will develop an appropriate user's model (UC), which allows them to use the system successfully. The conceptual design approach therefore introduces 3 models of the system users are interacting with:

(1.1) **User's model (UC)**[18]

An individual user's internalised representation of a specific computer system. The model is therefore instantiated and will usually be specific and descriptive. It is dynamic since it is assumed to change with user-system interaction, and likely to be distributed rather than structural for most users.

(1.2) **Design model (DC)**

An externalised, general design model of a specific system, produced by the designer. This will normally be structural, descriptive and static.

(1.3) **System image (C+MC)**[19]

The way in which the design model is presented to the user, through the user interface, documentation, instruction material, error and help facilities. It contains parts of the system, which is an expression of the design model, rather than a model in itself; models may, however, be provided as part of the accompanying materials and documentation (M).

(1.4) **Conceptual Model (UC+DC)**

Norman (1986) states that UC and DC form the conceptual model, without elaborating its characteristics or functions further. Many authors still use the term this term instead of design model (DC).

Norman (1983) also introduced a distinction between models held by users and designers, and the conceptualisation of these models by others – usually designers and researchers. Three conceptualisations have been mentioned:

---

[18] The numbering scheme for the models reflects the way the models are grouped in the summary list of models in Appendix 1 and is provided to aid cross-reference between the Appendix and this chapter.

[19] Nielsen (1990) assigns the label MC for system image. Since M is for Manual and other documentation, it could be argued that it does not cover undocumented parts so of the system which the user interacts with, such as the screen display. C+MC expresses Norman's (1986) notion of system image more precisely.

(1.1.1) **Designer's model of users' model (D(UC))**[20]

An internalised or externalised structural, generic representation of the user's model, which is probably descriptive rather than analytic, specific rather than general, and (like all conceptualisations) static. Norman (1986) defined UC to cover both UC and D(UC), but a distinction between the generic and the instantiated model should be made.

(1.1.2) **Researcher's conceptualisation of users' model (R(UC))**

A externalised, structural, generic model of the user's model, which is static, general rather than specific, analytic rather than descriptive.

(1.2.1) **Researcher's conceptualisation of the design model (R(DC))**

The researcher's view of the conceptual model may be different from the designer's model. It is likely to be externalised, structural, generic, static, and analytic rather than descriptive.

The mechanism of conceptual design has appealed to system designers, but attempts to put it into practice have not always been successful:

> *"The function of the designer is to communicate the design model accurately via the system image. The function of the user is to form a user model that bears as little resemblance to the design model as is humanly possible. The user will add to what is so clearly communicated on the display every past experience they have had, relevant or not. If the interface carries any trace of ambiguity, the user will find it out and jump to the wrong conclusion. In short, the user will insist on doing everything wrong, wrong, wrong!" (p. 130)*

In this quote, Tognazzini (1991) points out that formulating a design model and communicating it through the system image does not necessarily result in the user's model and user behaviour intended by the designer. Firstly, other factors – users' previous knowledge and experience – influence the model-building process. Secondly, constructing a "good" design model and communicating it through the

---

[20] The term conceptualisation seems to be applied to representations which researchers construct of models. The designer's model may be constructed in a less deliberate and analytic fashion, hence the term "model" is used rather than "conceptualisation".

user interface is not necessarily a straightforward process. Norman (1983) states that a design model should be an *accurate, consistent,* and *complete* representation of the system. In the revised version, Norman (1986) gives more specific instructions as to how the model should be constructed:

> *"Ideally, the model is based on the user's task, requirements and capabilities ... [and] must also consider the user's background, experience and the powers and limitations of the user's information processing mechanisms." (p. 47)*

Similarly, Tognazzini (1991), who subscribes to conceptual design approach despite the display of comic exasperation in the previous quote, suggests that designers should construct simple design models which:

(1) reflect *users' tasks*, rather than the underlying hardware and software of the system;

(2) correspond to *users' experiences and expectations.*

To follow these recommendations, we need to add three further sets of representations to the model-building process: users' tasks, general knowledge and experience, and information processing mechanisms.

(2.1) **User's Task Model (UT)**

An internalised model of a task to be performed. This is an instantiated, specific, descriptive, dynamic model of the task. Whether the model is structural or distributed will to depend on the complexity of the task and user's level of expertise. We can expect users with similar training and experience of the task to have similar models.

(2.1.1) **Designer's Model of Users' Task D(UT)**

The designer's generic model of the UT, which can be internalised or externalised. The model is likely to be specific, descriptive and static. Whilst approaches such as conceptual design prescribe that the design model should be based on the user's model of the task, there is a danger that the designer may start from their own model of the task (DT), rather than model the task from the user's viewpoint (D(UT)).

(2.1.2) **Researcher's Conceptualisation of Users' Task (R(UT))**

The analysis of the users' task can be conducted using a formalised method or tool (task analysis), and the resulting model will be externalised using some notation. Depending on the method or tool used, the model may be descriptive or analytic. Task analysis methods would claim that they model the users' model of the task, but if they are applied to the task as a purely analytical exercise, without verification by users, the result will be an R(T) rather than an R(UT). Task analysis methods aim to produce generic and general models, and are unlikely to reproduce the idiosyncratic and distributed features (including gains and inconsistencies) of an individual user's task model.

(3.1) **User's Existing Knowledge (UW)**

This is the background knowledge an individual user brings to the task, and which may influence the way in which they interpret information about the system and interact with it. This is a highly individual, internalised model which differs from UT in that it is general rather than specific. It is likely to be distributed and dynamic.

(3.1.1) **Designer's Model of Users' Knowledge (D(UW))**

Designers are likely to form an internalised generic model of the subset of users' background knowledge and experience which may be relevant to the task and system. This model is likely to be distributed and descriptive. As with the designer's model of users' tasks (D(UT)), there is a possibility that the designer's own model of the world (DW) might be used instead of the user's (D(UW)).

(3.1.2) **Researcher's Conceptualisation of Users' Knowledge (R(UW))**

This would differ from D(UW) in that it is likely to be externalised and analytic.

Since the design model is not communicated directly, but via the system image, the construction of the system image is as important as the construction of the design model itself. Norman (1986) prescribes that the system image should be *explicit,*

intelligible and *consistent*; the designer should realise every interaction contributes to forming the user's model. Norman acknowledges that this places a large burden on the designer. Tognazzini (1991) offers some practical guidance on designing the system image:

(1) Design user interface objects which encourage and facilitate user behaviour that is consistent with the design model.

(2) Do not use abstract or invisible objects.

(3) Remove any elements not needed for a specific task from the user's sight.

(4) Minimise the amount of information users have to remember.

These recommendations are presented as design heuristics, i.e. the craft knowledge described by Long & Dowell (1989). They are, however, directly based on Norman's (1986) theory, and like Norman, Tognazzini (1991) makes explicit reference to established knowledge about the powers and limitations of the human information processing mechanisms. We therefore have a further set of models – models of users' perception, memory, cognition and motor skills – to include in the process of constructing the system image:

(4.1) **Researcher's Model of Users (RU)**

Research in psychology and related disciplines has produced a number of theories and models of the working of the human perception, memory, cognition and kinaesthesia. These models are externalised, structural, generic, general, analytic and static.

(4.2) **Designer's Model of Users (DU)**

Designers might be using parts of RU, but are unlikely to be familiar with RU in its entirety. To follow Tognazzini's (1991) recommendations, they might consult a researcher to select or interpret a relevant part of RU for them. If this is not the case, they might draw on a their own, internalised version of RU, which is likely to be distributed, generic, and descriptive.

> *"The desktop metaphor ... is an inviting metaphor that provides easy access to the system. Once users are immersed in the desktop metaphor, users can adapt readily to loose connections with physical situations – the metaphor need not to be taken to its logical extremes." (p. 3)*

This statement implies that:

(1) evoking a metaphor that users are familiar with facilitates access to the system;

(2) a metaphor need not match the design model completely and in detail to be effective;

(3) the use of a particular metaphor will lead to a specific user's model.

These are strong claims which require a closer investigation of metaphors and their relationship to design models and user's models. According to Wozny (1989), both analogies and metaphors provide a mapping between two domains. An *analogy* (first introduced in 3.3) provides an explicit, referentially isomorphic mapping between objects in two domains; valid analogies can therefore only be constructed between similar domains. A *metaphor* is a looser type of mapping, which points out similarities between two domains, without making explicit links between individual objects, or involving all objects. A metaphor is therefore referentially arbitrary and open-ended; its primary function is to initiate a process of active learning in the user (Carroll & Mack, 1985).

Having distinguished the two types of mapping, we need to consider what the two domains involved in the mapping processes are. For an analogy, the domains need to contain similar objects. Wozny (1989) therefore suggests that in HCI, an analogy is used when users map knowledge about one computer system onto another. Thus, analogy can only be constructed for, or employed by, users with existing working knowledge of a similar computer system. An example would be describing a new spreadsheet application by reference to an existing one.

In the absence of knowledge suitable to construct an analogy, a designer would have to employ a metaphor to cue existing knowledge from a different domain. We can represent these two different mapping processes as follows:

(M2) **Analogy (UC1 → UC2)**

Mapping an existing user's model of one computer system onto a new system. This requires, naturally, that the user holds a similar model which can act as source of the mapping. The design model and system image would incorporate the existing user's model. The mapping would be expressed in the system image, and the user would develop a new model through interacting with the system (D(UC1) → DC → C+MC → UC2). Alternatively, UC1 may be evoked and amended through instruction. Wozny (1989) suggests that with increasing knowledge about computing, more users will be able to use analogies when learning a new system.

(M3) **Metaphor (UW → UC)**

Mapping an existing model from user's knowledge onto the a new system. This requires identification of a suitable model, incorporating it in the design model, and communicating it via the system image. In interaction with the system image, the user then develops a new model (D(UW) → DC → C+MC → UC).

Wozny (1989) describes the mapping of a metaphor as a three-stage process. In Stage 1, users explore the computer system guided by their knowledge of the source domain. In the course of the exploration, they identify which parts of the domain apply to the computer system and which do not (Stage 2). Identification of similar and different elements in the two domains provides the basis for the user's model, which is formed at Stage 3, and used to formulate and test predictions about the system. So far, the description of metaphor construction supports the claims made in the Apple User Interface Guidelines. However, Wozny (1989) points out that there are points of failure at every stage which may prevent a successful transition from metaphor to user:

(1) Infrequent (novice and casual) users may never get beyond relating their existing domain knowledge to the system (Stage 1), and will continue to reason about the system in terms of the source domain. These users will continue to use the system in a trial-and-error fashion. The difference between an intended metaphorical interpretation and the resulting literal

one is a pervasive problem in HCI (*viz.* the mechanism of *linguistic cueing*, discussed in 4.2.5) which interferes in the model-building process.

(2) Inconsistent or mixed metaphors may make it difficult for users to work out which objects, rules and constraints of the source domain apply to the system and which do not (Stage 2). Some users (especially those with low levels of experience and confidence in their abilities) are likely to react confused and dismayed, rather than shrug this off in the manner suggested in the quote from the Apple User Interface Guidelines. The desktop metaphor itself is, as Wozny (1989) points out, a *mixed metaphor*. It merges objects from the physical world and knowledge about system functions in one model, rather than providing a mapping between the two (D(UW + DC) → C+MC → UC).

(3) If the system has a significant number of features which have no equivalent in the real world (and are therefore not covered by the source domain), users may experience a *metaphor breakdown*[22] after initially successful interaction: they lose confidence in the metaphor before user's model has been established. Metaphor breakdown and subsequent complete abandonment of the metaphor by users is qualitatively very different from the cheerful *"If it fits, use it; if it doesn't, don't"* strategy suggested by the Apple User Interface Guidelines.

To summarise, the use of analogies to construct users' models seems to be restricted to users who already have a knowledge of similar systems. For users who do not have such knowledge, a designer could try and to cue a suitable metaphor from a different domain as a starting point. The process of constructing a user's model via a metaphor, is, however, a somewhat fragile process, which may or may not succeed.

---

[22] The term *metaphor breakdown* is chosen as an analogy to Winograd & Flores' (1986) *communication breakdown*. The latter occurs when the dialogue between a user and an anthropomorphic system image (one which suggests an analogy between human-system dialogue and human-human dialogue) fails.

4.2.3 Surrogate models, task-action mappings and distributed models

Both Young (1983) and DiSessa (1986) describe models which are mental analogues of the system the user is interacting with: surrogate models.

(M4) **Surrogate models (C → UC)**

> Users who hold a mental analogue of the system they are interacting with can predict system behaviour accurately, since they "run" a simulation of the system itself when using that model. These internalised, instantiated and specific models may be acquired through prolonged and regular interaction with the system. DiSessa (1986) suggests that users develop a functional view of the system to start with, then develop a sense for the structure underlying the functions, and finally discover the true amount of functionality (DC → MC+C; (UT)+(UW)+(MC+C) → UC1; UC1+(MC+C) → UC2).

Most users, however, want to complete tasks without having to construct a mental analogue of the computer system. Task-action mappings take user tasks as their sole starting point. The core of the mapping between task and system provides an overall characterisation of the system to orient the user's behaviour. Young (1981, 1983) argues that task-action mappings are a good basis for design: the overall shape of the mapping can be used to guide the designer – i.e. the task-action mapping becomes the design model. With complex systems, submappings will be necessary – once this happens, the relationships between the submappings become crucial. If submappings overlap, and one task can be located in two or more parts of the system, it is important to preserve consistency. The designer must ensure that the same task is translated into the same action in any part of the system, no matter which route is chosen: for instance, the task *"save the file I am currently working on"* should be mapped onto the same action in any part of the application. Failure to observe this principle would be likely to result in *mode errors* (Monk, 1986). With his statements on task-action models, Young (1983) provides the most specific and detailed prescription of "how to construct a design model".

(M5) **Task-action mapping (UT → UC)**

> Reduced models which account for the functionality of a system by reference to the real-world task the user is performing are called *task-action* mapping

models. Task-action mappings model a mental representation of a direct connection between a task and the user actions required to perform that task. In order to construct such a mapping the designer would select a minimum set of task-action relationships (the *core* of the model), and channel the remainder of the functionality through these core mappings (D(UT) → DC → C+MC → UC).

His proposal, however, has been challenged by both Tauber (1988) and DiSessa (1986), who raise doubts about the suitability of task-action mappings as the basis for the design model. DiSessa (1986) states that designers should aim for a surrogate model, and only count on functional models and the user's *distributed models* – i.e. the previous knowledge and background users bring to the task – in the earlier stages of learning (as described in M4).

Tauber (1988) also challenges the claim that task-action mappings can model the entire functionality of a system adequately, without any reference to surrogate models of the internal workings of the system. He argues that a pure task-action design model will always hide important aspects of the system's functionality, and hamper the user's performance. His prescription is to derive design models from formal descriptions of tasks and users:

(M6) **Formal Task-based Model (R(UT)+R(U) → UC)**

> Tauber (1988) proposes that design models should be based on formal models of the user's task; models of human cognition which are based on theory and validated by experiments should be added to the process (D(R(UT)+R(U)) → DC → C+MC → UC).

The formal system suggested for modelling the task is TAG (Payne & Green, 1986); however, TAG is used to model system tasks (RC) rather than user's real-world tasks (R(UT)). The distinction and relationship between system and real-world is explored in more detail in 4.2.4.

4.2.4 Users' models and task models

The importance of the relationship between external, real-world tasks which users seeks to perform, and the computer system's functions which they have to master in

order to complete a real-world task, had been emphasised in early research on users' models by Young (1981) and Moran (1983). Green (1990) introduced the term *viscosity* to illustrate this relationship: the viscosity of a system increases with the number of internal (system) tasks the user has to know in order to perform the desired external (real-world) task. The number of translations the user is required to make is related to cognitive load. Payne et al. (1990) formulated the *Yoked State Space (YSS)* hypothesis to elaborate the relationship between users' models and task-action mappings. YSS states that:

(1) A user of any system must construct and maintain at least two separate state spaces, a *goal space* and a *device space*, and a *mapping* between the two.

(2) The *goal space* represents the possible states of the external world that can be manipulated with the device.

(3) The *device space* represents the possible states of the computer system.

(4) *Device operators* allow the user to transform states in the device space.

(5) A *semantic mapping* relates the entities in the device space to those in the goal space, so that device space states may represent goal space states.

Goal space, device space, and semantic mapping are all mental constructs. If the user wants to accomplish a transformation in the goal space (i.e. a task such as removing a paragraph), she has to apply operators in the device space (such as "marking", "block of text", DELETE, in a text editor). These state spaces (between external and internal tasks) need to be *yoked*. With the formulation of YSS, the need to distinguish between two different models of tasks involved in the process, which emerged in previous section, has been made explicit. The user's model of the task (UT)[23] needs to be decomposed into:

---

[23] YSS is sometimes referred to as a model of the user. It is a model of the user in that it is a conceptualisation of how users represent tasks, but at the core is the task representation.

(2.2) **User's Model of Real-World Task (U(WT))**[24]

This is the user's model of the task the user wants to perform in the real-world, i.e. a model of the primary task. This is an internalised, specific, instantiated, descriptive and dynamic model of the task. It is related to the concept of *goal space* of Payne et al. (1990), which is a conceptualisation of this model.

(2.2.1) **Goal space (R(U(WT)))**

Researcher's conceptualisation of how users represent a real-world task, which differs from U(WT) in that it is externalised, generic, analytic and static. As with (R(UT), depending on the method used to derive this model, the result may be an R(WT) rather than an R(U(WT)).

(2.3) **User's Model of System Task (U(CT))**

The user's model of the operations required to be performed on the system to complete U(WT). This is therefore a model of the enabling task. As a user's model, it will be internalised, specific, instantiated, descriptive and dynamic model; whether it is surrogate or distributed will depend on the user's level of expertise with the task.

(2.3.1) **Device space (R(U(CT)))**

Researcher's conceptualisation of how users represent operations required to complete U(WT). It is an externalised, generic, analytic and static version of U(CT). Some methods used to analyse system tasks are R(CT)s rather than R(U(CTs)).

(M7) **Semantic mapping (R(U(WT) → U(CT)) → UC)**

The mapping relationship proposed by YSS (Payne et al., 1990) states that the U(WT) is mapped onto U(CT) via function called semantic mapping

---

[24] A further modification of the notation has been contemplated at this point, since it is arguable whether the world can hold a model of the task. Possible alternative expressions to distinguish the sub-categories of UT would be U(T-W) and U(T-C). On the other hand, the system can hold an idiosyncratic model of the task, so U(CT) seems acceptable. The alliance of consistency and simplicity won over precision in this case.

(R(U(WT) → U(CT)) → UC). Semantic mapping is a proposed cognitive mechanism – a general RU – applied to a specific problem.

According to YSS, a user's model (Payne et al. (1990) use the term *device model*) provides a set of conceptual entities (the device space) and their inter-relationships, and a mapping from these entities into the goal space. In addition, the model may contain some elaboration of the device space that do not map onto goal states, but give autonomous semantics to steps in methods. In the relationship between a user's model and a task-action mapping, there are two further issues to consider:

(1) The user's model determines the parameters of the task-action mapping procedures. This, according to Payne et al. (1990), is important because the construction of such parameters (e.g. strings in text editing) means a major cognitive load for the user, which is often overlooked in theories of users' knowledge.

(2) The hallmark of an elaborated user's model is the ability to decompose procedures into meaningful components, which can be used to construct new procedures and identify shortcuts.

YSS, therefore, provides an explanation of how, in distributed users' models, surrogate and task-action mappings might be combined. It acknowledges the importance of model building blocks (such as strings and data types) and linguistic cueing mechanisms. Unlike the theories discussed in the previous subsections, Payne et al. (1990) do not prescribe or suggest how design models (DCs) should be constructed: its focus is providing a more detailed description of user's models (UCs). YSS is a researcher's conceptualisation of users' models R(UC), based on researcher's models of tasks (RT) and users' cognitive mechanisms (RU). Payne et al. (1990) point to parts of the design – especially labels – which have been shown to impact user's models and performance, and we can infer that the design model and the system image should support the semantic mapping from a task model to system function. In that, YSS does not differ from task-action mappings. The difference lies in the perspective on the mapping mechanism function: for Young (1983), the mapping is a prescription of how the design model (DC) and system image (C+MC) should be designed to support the construction of an appropriate user's model (UC). Payne et al.'s (1990) focus is the structure and content of the UC,

and then seek to illuminate the process through which it its constructed. Even though his work is not cited in this paper, their aims and methods show similarities to Johnson-Laird's (1983) approach. Not surprisingly, the review of the final theoretical contribution, which is directly based on Johnson-Lairds's theory of mental models, highlights some of the same mechanisms as YSS.

4.2.5 User's models as mental models

Manktelow & Jones (1987) provide the only contribution which explicitly aims to apply Johnson-Laird's (1983) theory of mental models, and related empirical research on memory, learning and thinking, to users' models and the design of systems.

The authors start from the same premise as Norman (1983, 1986) and other research reviewed in this volume: that user performance depends on having a single, coherent, plausible[25] model and accessing it at the right time. Manktelow & Jones (1987), however, supply a more detailed definition of these properties:

- *single* meaning that the user refers to one and only one model, rather than considering several potentially relevant models at the same time;
- *coherent* being defined in terms of *co-reference* (each proposition referring to an entity referred to or defined elsewhere in the model) and *consistency* (all properties and relations being compatible with one another and free from contradiction);
- *plausible* meaning that the user's model is part of an appropriate framework, i.e. knowledge about (a) similar systems and procedures, (b) the problem domain, and (c) the language used to describe it.

Following Johnson-Laird (1983), Manktelow & Jones (1987) state that there are two ways of in which such a user's model can be constructed:

(1) by cueing an existing model and extending it; or

(2) by construction *de novo*.

---

[25] Norman and other literature based on his work use *appropriate* rather than *plausible*.

Unlike the conceptual design approach (see 4.2.1), Manktelow & Jones (1987) spell out that these two processes require different types of support, which has important implications for designers trying to facilitate the construction of a certain user's model.

(1) Cueing and extending an existing user's model

An existing user's model is cued through the design model/system image and used as an *advance organiser* for new information. New information presented to the user is encoded as propositional representations, which are interpreted with respect to the existing model and integrated into it. User action is guided by the existing model, which can be scanned in any direction. This process obviously requires that users have an appropriate model which can be cued. An appropriate existing user's model is (a) of similar structure to, and (b) linguistically compatible with, the design model. Even though the authors do not use the term in this connection, this requirement suggests that the process of cueing and utilising an existing model is similar to an analogy ((D(UC1) → DC → C+MC → UC2) – see 4.2.2).

(1a) Structural similarity

Structural similarity does not assume formal equivalence. Manktelow & Jones (1987) emphasise that the *formal equivalence fallacy* – that users will match models on the grounds of similar formal (logical) structure – has been substantiated by Johnson-Laird (1983). Such reasoning mechanisms may be applied by users trained in this manner, but it is unlikely to apply to the majority of users. Structural similarity depends on content rather than form, i.e. users need to be reminded of a general class of knowledge in which the propositional representations can be accommodated.

(1b) Linguistic compatibility

Linguistic compatibility means that the propositional representations to be integrated into the existing model are presented (and therefore encoded) in a form which allows the user to identify equivalent terms in the existing model and the propositional representations. Linguistic compatibility provides a different perspective on Wozny's (1989) insistence that analogies can only

exist between similar domains: it is the similarity of vocabulary which facilitates analogy, rather than identical formal structure.

(2) Constructing a new model

Without an existing user's model into which new information can be integrated, propositional representations remain in the user's working memory and direct users' actions. Since propositional representations are encoded as raw data (i.e. in the format in which they have been received in memory), they can be of arbitrary syntactic structure and are not integrated. When interacting with the system on this basis, users will select what they see as the appropriate propositional representation and act on it. This selection process is driven by heuristic processes. A designer trying to assist the process of construction *de novo* has to consider (a) the number of propositional representations to be processed by the user; and (b) the form in which they are presented.

(2a) Number of propositional representations

Juggling even a small number of propositional representations poses a considerable load on the user's working memory, which means that memory space required to construct a user's model does not become available. With a new model of any complexity, users will require the opportunity to evaluate a subset of propositional representations and establish relationships between them. Once these are consolidated, they form the starting point for a model and free up space in working memory, allowing a new batch of propositional representations to be introduced, processed and integrated with the basic model. Hence, knowledge presented to the user needs to be partitioned and introduced in stages, providing opportunity for practise and consolidation.

(2b) Format of propositional representations

Since available propositional representations are encoded in their raw data format, the format in which knowledge is presented is important. One requirement is that propositional representations should be linguistically consistent within themselves. Secondly, propositional representations need to be constructed to match the heuristics by which users work, such as their

linguistic biases. Designers can reduce the processing overhead and error rate by (a) exploiting matching bias; and (b) avoiding use of negatives, marked words and suppositions. Manktelow & Jones (1987) also emphasise the importance of consistency with salient task features (presumably of the real-world task) as an important heuristic; this may suggest a principle similar to a task-action model (see 4.2.3), but this is not discussed in sufficient detail.

Manktelow & Jones (1987) summarise their findings in a set of 24 Summary Design Principles (see Appendix 2). In content and form, these correspond to the engineering principles advocated by Long & Dowell (1989): they are (a) based on integrated empirical evidence, (b) prescriptive, and (c) testable statements about effects of design and instruction on users' models and performance. The 24 principles and 13 sub-principles are fairly complex, and presented in scientifically cautious terms, rather with the authoritative air of a successful designer which accompanies Tognazzini's (1991) list of recommendations (see 4.2.1).

Whilst Manktelow & Jones' (1987) Summary Design Principles may be less accessible, they do correct a number of misconceptions and fill gaps apparent in the conceptual design and metaphor-based approaches. Available evidence – both from general research on cognition and HCI (e.g. Briggs, 1988) – suggests that the structure of internalised models, such as users' models and their existing knowledge and experience, is very different from the structure of externalised models used by designers and researchers. Users' knowledge structures are highly individual and unique; yet the assumption of the metaphor-based approach in particular seems to be that the user's model cued will correspond to the externalised, ideal version envisaged by the designer. Johnson-Laird (1983) debunked the notion that human reasoning follows the line of formal logic, yet a number of specific inference mechanisms are tacitly assumed as part of the process of transferring an externalised ideal model into the user's mind. Conceptual design and metaphor-based approaches aim to facilitate construction of a user's model, but have not paid attention to the implications of procedural semantics. The capacity of a user's working memory has to be considered in the construction of models, and the format of verbal instructions plays a crucial role in this context. Like Payne et al. (1990),

Manktelow & Jones (1987) point out that if designers aim to facilitate the construction of a user's model, they have to start by facilitating the construction and integration of the building blocks.

4.3 Empirical contributions

Empirical studies of users' models of computer systems have been published since the mid-eighties. The review in this section concentrates on studies of computer systems and commercial software, since it is the application of mental models to the design of such systems which is the focus of this thesis. Computer systems are different from most other devices because they can be used for a multiplicity of tasks (see 4.2.2). Therefore, studies of less complex devices, such as calculators (Young, 1981; Halasz & Moran, 1983, Bayman & Mayer, 1984) and cash dispensers (Payne, 1991a), are not reviewed in detail in this section, but findings which are relevant to users' models of computer systems are incorporated in the conclusions (4.4.3)[26]. Similarly, studies of user knowledge and behaviour which make no explicit reference to users' models (e.g. Payne, 1991b; Mayes et al., 1988) are not reviewed in detail here, but will be drawn on in the conclusions. A summary overview of the studies reviewed in detail in this section is presented in Table 7.

The purpose of the critical review presented in this section is to sift the empirical evidence to establish:

(1) what *types of models* have been studied;

(2) whether the results *support* or *contradict* any of the theoretical contributions presented in 4.2;

(3) which *methods* have been employed to investigate users' models.

---

[26] The study by Kieras & Bovair (1984) is the one exception – it uses an artificial control panel device which has only one purpose. Since this device has a certain level of complexity, and the study is much cited, it has been included in the detailed review.

Study	No of subjects	Users	System	Models and mappings	Investigative method	Data
Kieras and Bovair (1984)	40 (2 groups)	novices (students)	control panel	surrogate model C → DC → UC	lab experiment training	task performance some verbal protocols
Borgman (1986)	28 (2 groups)	novices (students)	on-line catalogue	metaphor UW → DC → UC	lab experiment training	task performance interview
Frese et al. (1988)	15 (3 groups)	novices (students)	word processing	surrogate model C → DC → UC1 construction *de novo* C + R → UC2	lab experiment training	command recall task performance
Briggs (1988)	6	novices (adult trainees)	word processing	construction *de novo* T+C+M → UC	lab experiment observation	verbal protocols task performance
Van der Veer & Felt (1988)	10	managers	integrated office system	metaphor R(UW) → UC1 surrogate model C → UC2	field study training observation	questionnaire observation interviews teach-back
Marchionini (1989)	16	novices (high school students)	electronic encyclopaedia	analogy UC1 → UC2	field study training observation	task performance keystroke logs verbal protocols interviews
Gray (1990)	10	novices (adult volunteers)	hypertext system	adaptation of users' models C → UC1→ UC2 → UC3	lab setting observation	user behaviour verbal protocols drawings
Payne et al. (1990)	Exp. 1: 14 Exp. 2: 24 Exp. 3: 48	novices and expert users novice users novice users	card sorting 2 word processing exp. text editor	device models R(U(CT))	lab experiment	task performance verbal protocols
Van der Veer & Wijk (1991)	54	novices (adult employees)	spreadsheet	metaphor DC → RW → UC	field study training observation	performance self-reports teach-back

Table 7: Empirical studies on users' models

A detailed review of the methods employed by three early experimental studies which investigated the role of the user's models – Borgman (1986), Briggs (1988), and Frese et al. (1988) – and a much-cited study using an experimental device – Kieras & Bovair (1984) – has been presented elsewhere (Sasse, 1991). The two main criticisms (*over-reliance on performance data* as an indicator for models or their characteristics, and *lack of ecological validity* of the findings) will be revisited in the conclusions of this section. Since the earlier review pre-dates the adoption of Nielsen's (1990) taxonomy, detailed reviews of those studies are included here.

4.3.1 Models of a Control Panel Device

In this widely-quoted early set of 3 experiments, Kieras & Bovair (1984) trained two groups of users in procedures which enabled them to operate a simple control panel device. In addition, users in the experimental group were provided with an externalised, surrogate model – a circuit-style diagram – of the fictitious device. The device is described as a phaser bank on Starship Enterprise of Star Trek fame, thus providing *"an interesting fantasy context"* for the task. The model was accessible to users during the training phase, but not during the benchmark tests. The authors report a significantly better performance of the experimental group – measures taken were *training time, time to complete task, ability to identify and correct inefficient procedures*, and *retention over time.* They attribute this to the effective users' models of the device formed by the experimental group as a result of exposure to the surrogate model. More precisely, the effect of the model is described as allowing users to infer specific procedures: verbal protocols from Experiment 2 showed that users in the experimental group made reference to the model when inferring procedures. The authors go to some length to eliminate alternative explanations[27], indeed the whole experimental procedure is very meticulous. Thus, Kieras & Bovair's (1984) results demonstrate the effectiveness of a surrogate model – the device used in the experiment was simple enough to allow users to acquire and maintain such a model in the time available – as opposed to rote learning.

---

[27] The possibility that the interesting fantasy context may have motivated users in the experimental group more than those in the control group, and hence cause the difference in performance, was eliminated in a subsequent experiment.

The authors propose that less precise models which do not support specific inferences – *"general principles, metaphors and analogies*[28]*"* – will be of little value since they are unlikely to support such specific inferences. This would support DiSessa's (1986) and Tauber's (1988) view that users need surrogate models for effective interaction. The somewhat dispiriting conclusion for users and designers is "no pain, no gain": there is no easy way of acquiring useful models of complex systems. This would, however, be an over-generalisation of Kieras & Bovair's (1984) findings. The conclusion that other types of models are not helpful is derived in an indirect fashion: alternative types of models and their effects were not investigated in the study. The generalisability of results obtained from short-term interaction with a comparatively simple, artificial device is doubtful.

4.3.2 Models of an On-line Catalogue System

Borgman (1986) also used a "model" *vs.* "no model" comparison of two groups of novice users, who were trained to use a prototype of an on-line retrieval system for a database of bibliographic references which was installed in a university library. The instructions for the experimental group explained the system through an analogy with a card catalogue, in addition to the set of procedures for retrieving literature which had been given to the control group. Task performance was measured through time taken to complete tasks, and number of tasks completed without errors. No significant differences between the groups were found on simple benchmarks tasks; but on more complex transfer tasks, the "model" group performed significantly better. These results would oppose Kieras & Bovair's (1984) conjecture that analogies are not effective in transferring knowledge.

Borgman (1986) tried to corroborate the findings from performance results by demonstrating the presence of different models in the two groups. The verbal descriptions of the system obtained from interviews with users, showed, however, showed that only 3 out of 14 subjects in the "model" group described the system in terms of the externalised model – but then, so did one of the users in the "no model"

---

[28] Following the definition used in this thesis (see 4.2.2), analogies have more in common with structural models than with metaphors and general principles, since both provide an isomorphic mapping relationship.

group. The author offers several explanations for the experimental group's failure to refer to the taught model. The first – that eliciting user's models may be more difficult than anticipated – is confirmed by other observations and discussed in some detail in the conclusions of this chapter (see 4.4.3). The second explanation offered – that users were so intelligent (Stanford University students, after all) that they did not need to use an externalised model, but preferred to construct their own – seems both laboured and pompous.

Nevertheless, the second part of the explanation – that users preferred to construct their own models - may be correct. Users in the experimental group may have failed to adopt the model because it did not support their interaction with the system. The model presented to them was *"an analogical model of the card catalog"*, chosen *"because of its obvious relation to the on-line catalog."* The relationship between a card catalogue and an on-line retrieval system may be obvious to a user who is familiar with the task for which both are tools, but a card catalogue is not analogous to an electronic retrieval system. An analogy provides an isomorphic mapping between two similar domains (see 4.2.2). Whilst both a card catalogue and an on-line retrieval system allow users to find literature by author, title and subject, not all features of the system can be explained in terms of the card catalogue. One of the great advantages of an on-line retrieval system – which users in this study were supposed to exploit – is its ability to handle multiple search criteria in one query, where use of a card catalogue would require a cumbersome sequence of find-and-match operations. This is confirmed by the statement that:

> *"The most difficult system feature to present in terms of a card catalog was the Boolean logic capability, as no direct analog exists." (p. 52)*

Because the mapping between the two domains was incomplete, the analogy would break down at some point (akin to the metaphor breakdown described in 4.2.2), and be discarded by the majority of users. Those who did refer to the card catalogue may have used the model as a metaphor or a task-action mapping. Users who were not very familiar with bibliographic searches and card catalogues to start with would not have such a model to be cued and amended, and would therefore try to construct their own model from the instructions. The fact that many users found the tasks difficult – 26% of recruited users failed an easy benchmark retrieval task –

suggests this possibility. Borgman (1986) concedes that, in retrospect, users' existing knowledge of the task domain should have been checked as part of the recruitment process.

4.3.3 Models of a word processing system (1)

The study by Frese et al. (1988) also tried to use different types of training to encourage the construction of different types of models. Three groups of users were trained to use a commercial word processing system. As in the two previous studies, the difference between the first two groups was training in procedures only (rote learning) *vs.* procedures plus design model (C → DC → UC1). The experimental group, however, was encouraged to develop what the authors call an *active user's model* by exploring the system. Whilst they received no instruction or training, they had access to an expert (researcher) throughout their exploration process. The researcher encouraged users to formulate hypotheses about the functioning of the system and put them to the test (C + R → UC2). The authors' hypothesis was that this process would lead to the construction of an active users' model, which would in turn manifest itself in superior performance on a transfer task (a complex task performed under speed conditions). In addition to performance on the transfer task, command recall, errors and efficiency were measured. Users were also asked to rate their own satisfaction with the training.

Given the small number of subjects in each group, the results have to be treated with caution, even though authors state that differences are significant[29]. The largest differences were found on the performance measures – time and efficiency. The experimental group performed significantly better than the "no model" group, and marginally better than the "passive" model group. The transfer task shows a similar pattern: the experimental group performed best and the "no model" group worst; the difference between these two groups is significant, whereas the difference between "active" and "passive" model groups is not. User satisfaction with the training was at similar levels in all 3 groups.

---

[29] The authors themselves admit *"we did not use very conservative statistical tests"*, but feel nevertheless that *"our general interpretation can be justified by the data, since all the performance results point in the same direction."* (p. 302)

The result adds to a growing body of evidence that a purely procedural training approach results in less efficient user behaviour and inferior performance on a transfer task (Kieras & Bovair, 1984; Borgman, 1986, Sasse et al., 1988). Frese et al. (1988) are convinced that this performance difference is due to a different structure of users' models. They propose that the use of hypotheses at the start of the training of the experimental group encouraged the development of *"some explicit cognitive preconception of the system"*; the knowledge acquired during user-system interaction is integrated into such a preconception, which is finally elaborated into a full user's model. This hypothesis of the model-building process exhibits certain similarities with the *advanced organiser* function described by Manktelow & Jones (1987, see 4.2.5). If "active" models constructed in this manner were superior to "passive" ones, potentially expensive training, rather than design, would be the way to induce users' models. The authors do not provide a corresponding description of the process by which externalised "passive" models are turned into user's models, but it is fairly clear that the presented models are assumed to be transferred more or less unchanged into users' minds. The fallacy of "passive" model construction, which is at odds with mental models theory (Johnson-Laird, 1983), and has also been denounced for users' models in HCI by Carroll & Rosson (1987), is discussed in more detail in the conclusions of this chapter (see 4.4.2).

4.3.4 Models of a word processing system (2)

Like Frese et al. (1988), Briggs (1988) investigated self-directed learning and its influence on users' models. In this small-scale study, 6 female users with no previous computing experience were presented with a word processing system and a task (editing and printing a letter). The users were encouraged to ask the trainer as many questions as they liked before switching the computer on, and were allowed to take notes. They were told that after switching the computer on, they were only to ask questions if they could not proceed without help from the trainer. The questions asked by the users were the focus of this study; they were recorded and analysed to determine what kind of information inexperienced users are looking for when faced with a new system.

Briggs (1988) observed that users' questions are very much directed by what they see: when presented with the task, their questions are driven by visible features of the task (e.g. how to underline or centre text); once the system is switched on, users ask questions about visible parts of the system (e.g. meaning of keys, menu items); these questions were asked in a fairly systematic and coherent manner. The number of questions about "how the system worked" (i.e. knowledge required for the user's model) increased once the system was switched on, but were fairly imprecise and not followed up by the users. Briggs (1988) attributes this to

> *"... the absence of a good [design] model of the system which can act as a framework or meta-structure for learning. Without this, users are forced to rely on available cues to guide the learning process, the most obvious of which are surface features of the task and the machine." (p. 440)*

Like Frese et al. (1988), Briggs suggests that users need something like an *advance organiser* (Manktelow & Jones, 1987) to facilitate the processing of incoming information. If no such advance organiser is given, users resort to their own knowledge and experience for the "next best" model from which to construct a user's model. Once cued, the model is disastrously resilient: rather than evaluating the model against incoming information, users bend new information to fit into the model.

The finding that users' information seeking is very much guided by what they see at the time endorses the notion that the system image can guide the construction of users' models (Norman, 1986). The absence of a design model at the beginning of the learning process adversely affects user-system interaction in the long term; yet, we know from other studies (Carroll et al., 1985) that users do not tend to make good use of instruction materials which aim to communicate such models. The reason may be, as Briggs (1988) observes, that users' information seeking strategy is task-driven, which would strengthen the case for task-action models (see 4.2.3). The other reason may be that whilst the model presented may be suitable, the process by which users are expected to acquire it is not. Briggs (1988) felt that users were *"ludicrously overconfident"*, asking on average only 7 question before switching the computer on. An alternative explanation can be derived from Manktelow & Jones (1987): users stopped asking questions because the information given in response

was as much as they could hold in working memory; they needed an opportunity to test and consolidate this information before acquiring more. This opportunity was denied to them in Briggs' (1988) study, and it is not catered for in most teaching and instruction materials.

4.3.5 Models of an office system

Van der Veer & Felt (1988) introduced 10 insurance company managers to an integrated office system in a 3-day course. All participants had used computers before, but were new to the integrated office system, which consisted of software modules (spreadsheet, database, text editor, graphics, communication and system configuration functionality). The system had a global command set (COPY, DELETE etc.) but the authors point out that, since the integrated system had been constructed from individual pieces of software, some syntactic inconsistencies existed for the use of these commands across the different parts of the package.

Users were introduced to the system in terms of a task metaphor and a visual-spatial metaphor. The *task metaphor*, which was introduced first, related the package to a company, with different parts of the package being related to functions performed by different people (draughtsman for graphics, accountant for spreadsheet, typing pool for text editor, etc.). Following the classification of models introduced in 4.2, this is not a model of the users' task, but a metaphor relating users' knowledge of a different domain to system functionality (UW → DC → UC). The *visual-spatial metaphor* represented the package through a drawing of a series of boxes connected by pipes. Van der Veer & Felt (1988) report that the drawing representing the entire system proved too complex for the users; therefore, an additional set of simpler drawings was added for individual software components. Given that these drawings represent parts and connections of the system in some detail, the mapping provided is that of a surrogate model (DC → UC), rather than a metaphor.

Van der Veer & Felt (1988) report that all but one of the users demonstrated what they consider a *"reasonable level of semantic knowledge"* about the software modules and operations which can be performed in them, and that 8 users had developed appropriate users' models. The models were elicited at the end of the training period by asking users how they would teach the software package to a new

colleague who wanted to complete a certain task. The representations elicited include:

(1) a list mapping objects to particular software modules;

(2) a list of procedures;

(3) an explanations in terms of the "draughtsman" and other roles;

(4) a flowchart;

(5) a keystroke representation;

(6) a drawing showing relationships between software modules.

The third clearly reproduces part of the metaphor, and the last shows similarities with the surrogate model presented to users during training. It may seem surprising that most users did not refer to these models, given that users were asked to explain the system with respect to a particular task which had been linked to the metaphor during the training. The authors do not comment on this finding. The users in this study had previous experience of similar systems; most of the representations, while diverse, are all related to computing knowledge. It is possible that users preferred to draw on their existing models from a closely related domain, and constructed their users' models of the new system by adapting models they were familiar with. Users in this study had sufficient time and opportunity to practise and consolidate information about the new system; this would have helped to integrate information about the new system with existing knowledge.

4.3.6 Models of an electronic encyclopaedia

Marchionini (1989) investigated 16 high school students' models of a paper-based encyclopaedia, and their effect on users' models of an electronic encyclopaedia. In a series of observations and interviews, comprising 3 sessions over a 7-week period, students were observed using a paper-based encyclopaedia to conduct an information search, before being introduced to an electronic encyclopaedia. In an attempt to provide meaningful and motivating tasks, students were allowed to propose their own search topic; those failing to propose one could select from a list prepared by the observer. Their search performance was evaluated in terms of search time, number of queries entered, and number of successful queries (obtained

from the observer's rating and keystroke logs). Qualitative data included interviews with students about their search strategies, and perceived differences between paper-based encyclopaedias and the electronic system. The interviews were audiotaped and analysed and coded independently by 6 researchers.

Marchionini (1989) reports that, using a generous definition of a successful search, 80% of searches in the printed encyclopaedia and 67% of searches using the electronic systems were successful. Search time was double for electronic than for the printed version; this is explained with students needing time to get used to the system, and higher motivation to persevere with the new system, compared to the book encyclopaedia. Even though 80% of searches were classified as ultimately successful, Marchionini (1989) reports that many searches were poorly formulated. 9 students made no use of Boolean search capability at all; 7 used the simplest connective, AND; the connectives OR or NOT were not used by any users. This observation on problems with Boolean search capability was also reported by Borgman (1986). Marchionini (1989) concludes that users have problems integrating these novel features into their existing model of the printed version. Following Johnson-Laird (1983), we would predict that users who were not trained in the use of abstract operators would have problems with in applying these. On the other hand, Marchionini (1989) also reports that only 4 out of 16 students made use of the index of the printed encyclopaedia, which suggests that most users' models of the printed encyclopaedia and of the search task were not very sophisticated. Similarly, Borgman (1986) was surprised by a low level of task competence among her users. Payne et al. (1990) would predict that users who do not have an adequate representation of the real-world task (U(WT)) will encounter problems in their attempts to construct a model of the system tasks (U(CT)).

From the verbal protocols, Marchionini (1989) concluded that all subjects used the print encyclopaedia as an *advance organiser*[30] (Manktelow & Jones, 1987) for their user's model of the electronic system. From intra-individual comparisons of performance and verbal protocols, he established that the more detailed the user's

---

[30] Marchionini (1989) uses the term *building block* rather than *advance organiser* to describe this concept, but this term could be easily confused with propositional representations.

model of the print encyclopaedia, the less they were able to accommodate novel features of the electronic system. He therefore proposes that, in order to further the construction of appropriate user's models, the instructors and/or instruction material (i.e. the design model and system image) should explicitly represent the differences between the existing user's model and the new system. This proposition ties in with Manktelow & Jones' (1987) Summary Design Principles 7-9 (see Appendix 2), and aims to reduce the cognitive overhead associated with adapting similar but not isomorphic models, such as metaphors (as discussed in 4.2.2).

An additional observation on cognitive load reported by Marchionini (1989) is that users preferred to use a small set of global operations, rather than using specific shorter commands – as Norman (1983) observed, they traded physical (executing longer command sequences) for mental effort (memorising additional specific commands). Limitations of users' working memory (Manktelow & Jones, 1987), or incomplete representation of real-world task (U(WT)), system task (U(CT)) and semantic mapping (Payne et al., 1990) would explain this observation.

4.3.7 Models of a hypertext system

Gray (1990) observed and probed 10 novice users exploring a hypertext system to determine their users' models, and how they changed over a series 10 search tasks. Users' models were inferred from behavioural and verbal protocols, and – using a new method to overcome the *model articulation problem* observed by Borgman (1986) – users' drawings of the way in which screens showing information were connected to each other. Drawings were collected after the first, fifth and final search tasks.

Like Marchionini (1989), Gray (1990) concluded that most users approached the system with models of linear text search strategies – i.e. a model of how they would search for information in a book. This impaired their ability to navigate the hypertext structure, and contributed to them "getting lost" in it. In contrast to Marchionini (1989), however, Gray (1990) found that users adapted their models. After 10 searches, 7 users' models showed a basic understanding of hierarchical structure of the hypertext system (drawing, or referring to, a "tree" or "web" structure), and half the models indicated an understanding of the cross-reference links. Thus, this study confirms the dynamic nature of users' models: clearly, users'

models change as a result of user-system interaction, at least in the early phases of learning about a system. Unfortunately, Gray (1990) does not discuss whether the results provided any insight into process by which the models were adapted. Thus, we do not know whether users discarded the linear model as it did not prove useful (in a process similar to that of *metaphor breakdown* as described in 4.2.2), and replaced it with a hierarchical model, or whether they evolved the linear model (e.g. through the process of procedural semantics), into a hierarchical user's model of the system.

An essential element of an appropriate user's model for successful navigation of a hypertext seems to be a representation of size of the system. With a book, users can easily determine the total amount of information available. Gray (1990) concludes that users require an indication of how much information is available, both in total and for a particular topic, and how much of it has been inspected. This may be an example for the basic type of information users require to construct a model as described by Payne et al. (1990).

Gray (1990) also observed a high number of incidences in which a process of linguistic cueing (Manktelow & Jones, 1987) led to user error. In 39% of searches, she detected that users attributed a different meaning to the search terms than those intended by the hypertext document designers. This high incidence of *category confusion* may be an indicator of users' lack of understanding of the domain. Unlike Marchionini's (1989) students, who were allowed to search for information on a topic of their own choice and were found to have a good understanding of the domains they chose to search in, Gray's (1990) users had to execute 10 prescribed searches in a specific domain (violent crime). Users' level of understanding of the domain was not checked prior to the study. Evidence from the partial transcripts provided in the paper suggests that the observed high incidence of *category confusion* may at least in part be due to lack of a sufficiently sophisticated vocabulary for, and understanding of, the specific domain. In this case, users would have to construct a model of the real-world task as well as trying to construct a users' model of the system. This would certainly represent an additional cognitive load (Manktelow & Jones, 1987; Payne et al., 1990).

4.3.8 Models of a word processor (3)

To test the YSS hypothesis put forward in the same paper (see 4.2.4), Payne et al. (1990) conducted a series of 3 experiments on the construction of device models (R(U(CT))). Rather than trying to induce such a users' model through instruction – a route taken by most of the other empirical studies – they observed the construction of a specific part of a device model (existence and use of copy buffers) over a series of tasks on word processors and text editors. The first experiment was a card sorting task, which demonstrated that expert users have a concept of a text string – an essential building block for editing tasks – whereas novice users do not. That absence of this concept does causes a significant cognitive load was confirmed in Experiments 2 and 3, where users consistently demonstrated problems inferring optimal procedures for editing tasks. In Experiment 2, 24 novice users were asked to perform editing tasks on two different word processing systems. The results confirmed the authors' hypothesis that the lack of the concept of a text string means most novice users are unable to infer optimal procedures for editing. Over a series of tasks, and with prompting from the experimenter, users did grasp the concept of a string more easily when using a mouse-based word processor (MacWrite), where marking of a string is perceived as a single action, then with a control-key based editor (IBM Personal Editor), where marking requires insertion of beginning and end markers. Once optimal editing procedures have been established, users are able to transfer these from the one word processing system to the another, but this transfer worked was more successful from the mouse-based system to the control-key one than vice versa. In Experiment 3, the authors investigated the effect of lexical names for operations on the device model by preparing different versions of a custom-built text editor. The results show that the label STORE produced somewhat better performance then COPY, and RECALL significantly better performance than PASTE. The authors draw two main conclusions from these experiments. Firstly, novice users require considerable time and effort to acquire basic concepts, such as text strings, and lack of such a basic concept impairs their ability to infer optimal procedures. The latter is taken as some indication of users' ability to construct a device model (U(CT)). Secondly, the acquisition of both basic concepts and device models can be influenced through the user interface, as

demonstrated by the superior performance following from a marking procedure perceived as a single action, and lexical names which are compatible with users' model of the goal space (U(CW)).

The performance results from these experiments support most of the authors' hypotheses derived to test YSS. Payne et al. (1990) conclude, however, that verbal protocols from the session indicate that successful user performance in such restricted task experiments should not be taken as conclusive evidence for presence of a correct model. They observed low levels of user confidence even when optimal procedures were correctly identified, expressions of surprise when such procedures worked. They conclude that

> *"... in some cases such restricted glimpses of performance may suggest false inferences about subjects' competence. By objective measures, subjects appear to have mastered aspects of the editor, but their talk reveals profound misunderstandings, which in real tasks could lead to difficulties." (p. 441)*

These observations confirm that performance results alone may not be reliable indicators of users' models, and make a strong case for using verbal protocols in mental models research.

4.3.9 Models of a spreadsheet

Van Der Veer & Wijk (1991) introduced 54 insurance company employees to a spreadsheet application as part of a field study. From documentation and instruction materials, they constructed a design model for the spreadsheet. On the basis of the design model, they constructed a number of *"functional, operational and organisational metaphors"*. These were presented to the users as part of a 2-day training course for the spreadsheet application, with the intention of facilitating the construction of appropriate users' models.

After the course, the authors attempted to elicit users' knowledge about the system – through a questionnaire and self-reports – and their models – through a *teach-back* scenario similar to the one employed in the spreadsheet study in Chapter 7. Whilst the size of the sample, type of subjects and real-world setting of the study are commendable and could have yielded ecologically valid results, it is disappointing

that only a few summary statements and 3 short examples are provided, without detailed discussion. Van der Veer & Wijk (1991) report that users' knowledge of the system was "mainly correct", and concluded that therefore the users had acquired effective users' models. Similar to Borgman (1986), they found that very few subjects described the working of the system in terms of the metaphors which had been presented. One metaphor for the spreadsheet represents menu (command) hierarchies as trees, files as apples, and spreadsheet macros as a caterpillar which eats the contents of a series of cells. Whilst this is an amusing and light-hearted manner in which to present system functions, it is an artificially constructed metaphor, which is neither task-based nor aiming to cue an existing model from users' general knowledge and experience. It is a mixed metaphor, combining knowledge akin to parts of a surrogate model with models from a very different domain. As such, it would be likely to break down and be abandoned by users (see 4.2.2).

4.4 Summary and Conclusions

4.4.1 Terminology

The term mental models has been liberally applied to virtually any representation of knowledge about devices (Tauber, 1988). Thus, it is understandable that HCI should have created its own terminology for a user's internalised representations of a computer system, and associated models. However, there is no single recognised set of terms which is universally employed. Norman's (1986) terms, along with his theory of conceptual design, are most often referred to in textbooks and research contributions, but are not consistently or universally applied. The analysis of terminology in this chapter identified aliases, impostors, and partly overlapping terms. The reader of a text often has to piece together what type of model has been presented or elicited from statements distributed across the paper; sometimes, it is not possible to infer the exact nature of the model at all. This state of the affairs may be explicable in that this is a new topic tackled by a researchers from different disciplinary backgrounds. It shows that HCI knowledge on users' models is far from being integrated – a state of affairs which Payne (1992) describes as *"confused and confusing"*, and which causes serious problems for both application and research.

Designers trying to apply HCI knowledge of users' models will be confused and irritated by diverse terminology and contradictory statements, and Newell & Card (1985) predict that this will lead them to ignore the topic altogether. The lack of a unified and universal terminology also poses a serious obstacle to integration and advancement of HCI knowledge on this topic. The review of empirical studies (in 4.4.3) confirms Nielsen's (1990) statement that comparing results from different empirical studies on users' models is the scientific equivalent of *"comparing apples with oranges"*. The result is a vicious circle: the lack of a unified terminology reflects the state of knowledge, but integration and advancement of knowledge is hampered by lack of a unified terminology. This circle needs to be broken if progress is to be made.

Nielsen's (1990) attempt to improve the situation by proposing a notation to describe models more precisely has not been taken up by researchers to date. This is regrettable, since the notation is simple, yet powerful enough to represent the models and mapping relations which have been proposed. Its application – with a few minor modifications and additions – in this chapter has demonstrated that applying the notation helps to identify similarities and differences between models, which the diverse terminology might otherwise obscure. Firstly, the notation clearly states who holds a model of what (see Table 8), which removes some existing confusion.

Who has model of	**User**	**Designer**	**Researcher**	
System	**UC** User's Model	**DC** Design Model	**R(UC)** Conceptualisation of User's Model	**R(DC)** Conceptualisation of Design Model
User		**DU** Designer's Model of the User	**RU** Researcher's Model of the User	
Task	**UT** User's Model of Task	**D(UT)** Designer's Model of User's Task	**R(UT)** Researcher's Conceptualisation of User's Task	
World	**UW** User's Knowledge and Experience	**D(UW)** Designer's Model of User's Knowledge	**R(UW)** Researcher's Conceptualisation of User's Knowledge	

Table 8: Models held by User, Designer and Researcher

Secondly, Nielsen's (1990) 6 dimensions help to identify certain characteristics and the level of abstraction of the different types of models (see Appendix 1 for a list of detailed descriptions). Table 9 shows an example how the dimensions could be used to develop a profile for a model which would provide a more precise description. As stated previously (see 4.1.4), the instantiations of any particular model may be located on different positions on some of the dimensions. Thus, it is not possible to provide a definitive profile for any of these models in general; rather, these dimensions should be used to describe a specific model proposed by a theory or detected by a researcher more precisely than is currently the case. Moreover, the exercise of identifying dimensions and describing instances of the phenomenon under investigation by placing them on such dimensions is part of *open coding*, a method which is part of the process of theory formulation in grounded theory (Strauss & Corbin, 1990).

Thus, the application of the notation could contribute more to the process of formulating theories of users' models than just the provision of unambiguous labels. Having added the symbol "→" – used in grounded theory to denote the existence and direction of a process – we can represent mappings between models, i.e. processes and dynamic aspects of user's models (see Table 10).

4.4.2 Theoretical contributions

The theoretical contributions surveyed in this chapter have a common goal: to provide a basis for the design of computer systems and user interfaces which support the construction of appropriate users' models. All theoretical contributions start from the premise that users form an internalised representation – a user's model – of a computer system they are interacting with. The content and structure of users' models direct user-system interaction. There is also agreement that the content and structure of a user's model is influenced by the information about the system which is presented to users, and how it is presented.

Models of: / Dimensions:		internalised – externalised	structural – distributed	generic – instantiated	general – specific	descriptive – analytic	static – dynamic
System							
User's Model	UC	internalised	distributed	instantiated	specific	descriptive	dynamic
Designer's Conceptualisation	D(UC)	internalised or externalised	structural	generic	specific	probably descriptive	static
Researcher's Conceptualisation	R(UC)	externalised	structural	generic	general	analytic	static
Design Model	DC	externalised	structural	generic	specific	descriptive	static
Researcher's Conceptualisation	R(DC)	externalised	structural	generic	general	analytic	static
System Image	C+MC	externalised	structural	instantiated	specific	descriptive	static
Task							
User's Model of Task	UT	internalised	structural or distributed	instantiated	specific	descriptive	static or dynamic
Designer's Model of Users' Task	D(UT)	internalised or externalised	structural distributed	generic	specific	descriptive or analytic	static
Researcher's Conceptualisation of Users' Task	R(UT)	externalised	structural	generic	general	analytic	static
User's Model of Real-World Task	U(WT)	internalised	structural or distributed	instantiated	specific	descriptive	static or dynamic
Goal Space	R(U(WT))	externalised	structural	generic	general	analytic	static
User's Model of System Task	U(CT)	internalised	distributed	instantiated	specific	descriptive	dynamic
Device Space	R(U(CT))	externalised	structural	generic	general	analytic	static
World							
User's Existing Knowledge	UW	internalised	distributed	instantiated	general	descriptive	dynamic
Designer's Model	D(UW)	internalised	structural	generic	general	descriptive	static
Researcher's Conceptualisation	R(UW)	externalised	structural	generic	general	analytic	static
User							
Researcher's model of user	RU	externalised	structural	generic	general	analytic	static
Designer's model of user	DU	internalised	descriptive	generic	general	descriptive	static

Table 9: Dimensional profiles of models (based on Nielsen, 1990)

The theoretical contributions diverge on the content and structure of user's models, and how exactly they influence the way in which users interact with a computer system. Not surprisingly, this difference in assumptions has led to a multitude of proposals as to the process through which mental models are constructed, and how this process could be influenced.

Structure and content of users' models

One of most striking results from the analysis of theoretical contributions is the lack of any examples of a generic or instantiated user's model – be it a "perfect" or a "faulty" one[31]. This raises the question of how designers should know what exactly they are aiming at. For most authors, the definition of an appropriate user's model seems to be an operational one: an appropriate user's model is one which works, i.e. which enables the user to interact successfully with the system. There are very few statements as to how the model influences user-system interaction; like the statements provided about content and structure of users' models these are at a general, fairly abstract level:

(1) Most individual users' models are incomplete, not "runable", without firm boundaries, unstable, "unscientific", and parsimonious (Norman, 1983).

(2) Users' models can be surrogates – mental analogues of the computer system – or task-action mappings between users' tasks and system functions. Expert users, who not only interact successfully with the system, but can also predict the outcome of operations and devise new procedures and shortcuts, hold a surrogate model. Most users' models, however, are assumed to be distributed, i.e. a patchwork of surrogate and functional elements connected to other knowledge (DiSessa, 1986). This explains the observations under (1).

(3) Users' models which are task-action models consist of a model of the real-world task (representing possible states of the task), a model of the system (representing relevant states of the system), and procedures for

[31] DiSessa (1986) provides an example model for a function in the programming language Logo, but this seems to be a DC rather than a UC.

mapping between them. User performance will therefore depend on the accuracy and completeness of the state representations, and whether the procedures can be decomposed to construct new procedures and shortcuts (Payne et al., 1990). Therefore, task-action models have a different content and structure from a surrogate model, but aim to support similar mental operations as those described in (2).

(4) A successful user's model is clearly identified, free from gaps and contradictions, integrated and linguistically compatible with user's knowledge about the task and similar systems, according to Manktelow & Jones (1987). They suggest that user's models are an instantiation of mental models as described by Johnson-Laird (1983), and work along the same principles.

When it comes to structure and content of users' models, there is agreement that users do form a model of a system they are interacting with, irrespective of whether they are encouraged to do so or not. The models which most users construct in this manner are less then perfect, and probably consist of procedures, task-action mappings, and parts of, or links to, other models which a user holds. In contrast to this are the surrogate models held by expert users, which are mental analogues of the computer system. Surrogate models take time and effort to acquire, and have to be maintained through regular interaction with the system. Thus, surrogate models are not attainable for casual users who just want to use the computer as a tool. Apart from DiSessa (1986), for whom a surrogate model is always the ultimate aim, the theoretical contributions assume that it is possible to induce a user's model whose content and structure are easier to acquire and maintain, but which is nevertheless structural (i.e. complete, free of contradiction, decomposable into procedures, and integrated with users' other knowledge). The structure has to be provided by *some guiding principle*, as Nielsen (1990) puts it, rather than replicate the structure of the computer system.

How users' models are constructed

One of the central tenets of conceptual design – building a design model and communicating it through the system image to influence the user's model – seems widely accepted. The main difference between the theoretical contributions is what

should form the basis of the design model and user's model. Table 10 summarises the different mappings which have been proposed for constructing design models and users' models. The user's task figures prominently in all mappings except for analogy (M2). The mapping between a new and existing system could form the basis of a design model (UC1 → DC → UC2), or be communicated through instruction only. A metaphor (M3) uses existing knowledge of the world (UW) as the basis for the design model; this may or may not include representation of a user's task.

	Theoretical Contributions	**Mapping**
M1	Conceptual Design Norman (1986)	D(UT)+D(UW)+R(U) → DC → C+MC → UC
M2	Analogy Wozny (1989)	D(UC1) → DC → UC2
M3	Metaphor Tognazzini (1991)	D(UW) → DC → C+MC → UC
M4	Surrogate Model DiSessa (1986)	DC → C+MC → UC
M5	Task-action mapping Young (1983)	UT → DC → C+MC → UC
M6	Formal task-based mapping Tauber (1988)	R(UT)+R(U) → DC → C+MC → UC
M7	Semantic mappings Payne et al. (1990)	R(U(WT) → U(CT)) → UC

Table 10: Proposed mappings for user's models

The task is the sole basis of the design model in task-action mapping (M5) and the more elaborate semantic mapping (M7), which distinguishes real-world tasks (U(WT)) from system tasks (U(CT)). Given that users' tasks are seen as the most important basis for design models and users' models, it is disappointing that no examples of users' tasks, and how they are to be mapped into the design models, are provided. HCI has developed a number of methods for task analysis, and some of these have made a successful transition into design practice (Diaper, 1989). It is disappointing to see that only Tauber (1988, see M6) makes an explicit suggestion that task analysis methods should be employed: his proposed method is a formal method (R(UT)), a predictive grammar. This is a type of method which, as Booth

(1991) points out, is neither sufficient to capture users' understanding of a task nor applicable by designers. Young's (1983) prescription for constructing a task-action mapping (M5) suggests a hierarchical breakdown of tasks (which most task analysis methods do provide), but does not state which guiding principle for breakdown should be applied (e.g. frequency or fundamentality of operations). The lack of pointers to suitable methods for identifying and describing users' tasks means that designers are left to their own devices; this lack of guidance almost certainly increases danger of a designer's model of the task (DT), rather than the user's model of the task D(UT) forming the basis of the conceptual model. Other sources for the design model (DC) include human information processing mechanisms (RU in M1 and M6) and users' knowledge and experience (UW in M1); there is, however, no prescription as to how these are to be integrated with task descriptions.

In summary, the theoretical contributions offer little concrete knowledge, guidance and support for a designer aiming to encourage appropriate users' models. It is not too difficult to work out how to construct an analogy (M2): identify an existing system that users are familiar with, make it explicit, and model or explain the new system by reference to the existing one. The model of the existing system will be cued and adapted into a model which describes the new one. A similar process would be applied for metaphor (M3), where the model cued would be part of the users' general knowledge, and the user would be left to work out the details of the new model from the point where similarities with the metaphor end. Conceptual design is more difficult to put into practise than analogies or metaphors: designers are to construct a design model which reflects users' tasks, corresponds to their experiences, and takes the limitations of human information processing into account. There are no prescriptions or even examples as to how knowledge of these can be turned into a design model. Tognazzini (1991) provides a few heuristics for the construction of system image, but they do not provide sufficient illumination of the process to guide designers.

4.4.3 Empirical studies

The review of the 9 empirical studies of users' models was conducted to establish which of the models and mappings identified in 4.2 have been investigated to date,

and whether the results of the investigations support any of the theoretical contributions. The third major objective was to survey the methods which have been used for investigating users' models and their effect on user-system interaction.

Support for theoretical contributions

One of most disappointing findings of the review is that the empirical work on users' models seems to have been conducted separately from the existing theoretical contributions. Apart from Payne et al. (1990), who conducted a series of experiments to test the YSS hypothesis presented in the same paper, none of the studies reviewed here was undertaken to test one of the proposed theories. The theories put forward ways of designing systems which support the construction of appropriate users' models. Most of the empirical studies, however, attempt to influence users' models of commercial software systems through instruction (Borgman, 1986; Frese et al., 1988; Van der Veer & Felt, 1988; Van der Veer & Wijk, 1991). Users are presented with secondary design models (DCs) which have been constructed post-hoc by researchers. The use of such secondary – instructional – design models does not constitute a test of theoretical contributions such as conceptual design, since the design models and system images of those systems were not developed on the basis of them. Rather, these studies explore the use of analogies and metaphors as instructional aids in the manner of the studies collected in Genter & Stevens (1983) (see 3.3). The remaining studies (Briggs, 1988; Marchionini, 1989; Gray, 1990) try to investigate users' models by observing users' interaction with systems and their performance on tasks. Again, we can only extract evidence for or against some of the specific statements in the theoretical contributions about users' models and their effect on user-system interaction from theses studies, rather than use them to evaluate any of the theoretical contributions as a whole.

The studies which attempted to influence users' models through instruction demonstrate that users presented with a model of a new software system perform better than users who are given procedural instructions only, especially on transfer tasks (Kieras & Bovair, 1984; Borgman, 1986; Frese et al., 1988). General theories of procedural and declarative knowledge would predict as much, but the effect is attributed specifically to users' acquisition of the externalised model (secondary DC)

they were presented with as a users' model. Only Kieras & Bovair (1984) made an effort to ascertain that the differences between the two groups were due to a difference in the users' model, and the fact that it allowed users to make specific inferences about the system. The study demonstrates the effectiveness of a surrogate model, but does not prove the authors' claim (also made by DiSessa (1986) and Tauber (1988)) that only surrogate models are capable of supporting specific inferences.

Borgman (1986), for one, reports superior performance on a transfer task by users trained with a metaphor. However, her attempt to confirm that the improvement in performance was due to the metaphor – by eliciting users' models – was unsuccessful. Similarly, Van der Veer & Felt (1988) and Van der Veer & Wijk (1991) found that only a very small number of users described the system in terms of the instructional design model. Two possible reasons have been put forward to explain users' failure to replicate an instructional design model (DC): (a) that users do adopt, but cannot articulate, the model presented to them; or (b) that users prefer to construct their own models. The "model articulation problem", as Borgman (1986) describes it, poses a methodological challenge for HCI research on users' models, which is discussed further in the review of methods (see below). The second explanation – that users prefer to construct their own models – would fit with mental models theory, provided that users can draw on relevant existing knowledge. Manktelow & Jones (1987) state that users who have existing relevant knowledge and experience aim to integrate new information with an existing set of models; hence, a design model which does not fit with existing knowledge and experience is less likely to be adopted. The diverse set of descriptions elicited from experienced computer users by Van der Veer & Felt (1988) supports that view, since they demonstrate that users drew on computer-related knowledge (e.g. state-transition diagrams, programming-style rules) to describe the new system.

Two further explanations can be added for why instructional design models have failed to surface in users' models: (c) metaphor breakdown; or (d) that instructional design models cannot be transferred into users' models. Novice users who do not have relevant knowledge and experience to draw on should display a greater tendency to adopt instructional design models. That users in Borgman's (1986) and

Veer & Wijk's (1991) studies failed to do so could be due to metaphor breakdown (Wozny, 1989). It is difficult to imagine a connection between Van der Veer & Wijk's (1991) apple-eating worm metaphor and any task that a spreadsheet might be applied to. Borgman's (1986) card catalogue "analogy" is related to the retrieval task, but suggests a mapping to an existing tool which does not have the functionality of the new retrieval system which users were meant to apply. When an instructional model breaks down, users either proceed to interact simply on the basis of procedures – in which case they should not demonstrate superior performance to a group which has been instructed in the use of procedures – or attempt to find another model from their own general knowledge and experience to fulfil the role of an *advance organiser* (Manktelow & Jones, 1987). Finally, we have to consider the possibility that instructional design models cannot be transferred into users' models. Carroll & Rosson (1987) pointed out that *"users' minds are not blank slates for designers to write on."* Users' bring existing knowledge and experience to user-system interaction, and the learning process is an active one in which models are constructed, not simply transferred. Only Kieras & Bovair (1984) can claim to have successfully demonstrated such a transfer of models, but since their study used a less complex, artificial device and a contrived task context, and users had no existing knowledge and experience to draw on.

Briggs (1988) reports that novice users' information-seeking behaviour is task-driven. This observation boosts the claims of the theoretical contributions which specify tasks as the main source of design models, and highlights the specific weakness of the instructional design models not taken up by users. Marchionini (1989) also observed that reference to a traditional encyclopaedia hampered users' performance with an on-line system. Rather than explicitly teaching a metaphor, he suggested the link between the existing and new tool through hands-on practise with the former before introducing them to the latter. The evidence from all 3 studies suggests that metaphors do not make good instructional design models. Instructional design models which try to encourage users by suggesting familiar tasks and tools can impair, rather than help, users to construct effective users' models. Unfortunately, the breakdown of inappropriate instructional metaphors will affect less experienced users, at whom much of the effort to provide a link with

familiar tasks and tools is directed, more than experienced users, who indeed seem to be able to "shrug off" inappropriate models and proceed to construct their own users' models. The failure of metaphors as instructional design models in these studies does not – as Johnson (1987) suggests - prove that they are inappropriate as a basis for design models and system images. Users in Gray's (1990) study were able to evolve an existing users' model of linear information structure – as in a book – into a hierarchical one which was more appropriate for a hypertext system. Manktelow & Jones (1987) and Wozny (1989) suggest that successful adaptation of an existing model into an appropriate users' model depends on whether similarities and differences between the existing model and the new system are clearly communicated; this requires signposting the differences and/or consistent feedback during training.

One respect in which expert and novice users do not seem to differ is the extent to which they depend on the system image. Briggs (1988) and Payne (1991a, 1991b) observed that user behaviour is very much guided by what is visible during user-system interaction. This confirms the importance which Norman (1986) and Tognazzini (1991) place on the system image. The relationship between design model, system image and user's model seems, however, different from the way described in conceptual design. Payne (1991b) found that even very experienced users of a system could not predict the exact outcome of very basic operations, or describe a complex system task (U(CT)) beyond the first sequence. Once users sit in front of the system, however, they complete the tasks in the manner we would expect from expert users. These observations suggest two things about content and structure of users' models:

(1) Error-free interaction observed in experienced users cannot be taken as an indication that those users have evolved a surrogate model. A surrogate model would be "runable" and therefore would enable users to think ahead and predict the outcome of actions. The observation casts doubt on DiSessa's (1986) conjecture that experienced users develop surrogate models with prolonged interaction.

(2) A user's model which allows competent interaction may not be the single, coherent, and plausible model envisaged by conceptual design. It seems

that the model does not include a representation of the system image itself, or of task-action sequences. It is possible that these results demonstrate that users cannot articulate models they have.

The studies by Payne (1991a, b) and Mayes et al. (1988) suggest that successful interaction observed by experienced users is very much due to perceptual stimuli and motor feedback. The theoretical contributions tend to treat users' models as purely cognitive phenomena, but these results clearly indicate that other mechanisms other than cognitive ones are at work.

At the same time, there is empirical evidence for the cognitive mechanisms which Manktelow & Jones (1987) suggest have a major influence on the model-building process. Both Briggs (1988) and Marchionini (1989) report observations which indicate that limitations of working memory interfere with users' information-seeking and model-building process. Payne et al. (1990) demonstrate that different lexical names have an effect on users' ability to identify effective procedures, whilst Gray (1990) reports that users' misinterpretation of labels used in the system hampered their performance and development of the model. Frese et al. (1988), Briggs (1988) and Marchionini (1989) describe the function of existing models in exactly the same way as suggested by the *advance organiser* described by Manktelow & Jones (1987).

Methods

Apart from looking for evidence for or against existing theoretical views of users' models, the review in 4.3 set out to identify suitable methods for conducting HCI research into users' models. The conclusion of an earlier review of the first 4 studies (Sasse, 1991) was that the experimental investigations suffered from a lack of ecological validity, and that authors tended to over-interpret differences in user performance data as indications of specific differences in users' models. The lack of ecological validity stemmed from:

(1) the use of artificial devices and tasks (Kieras & Bovair, 1984);

(2) the focus on novice users in the very early stages of learning only (Borgman, 1986; Briggs, 1988; Frese et al., 1988); and

(3) the small numbers of users involved (Briggs, 1988; Frese et al., 1988).

The more recent HCI studies of users' models all study commercial software systems, or parts thereof, rather than artificial devices. The tasks studied are real-world tasks (U(WT)) or system tasks (U(CT)) which are appropriate to those system. There is still a tendency to study first-time users of systems, but the research has been broadened by studying users who had previous experience of other computer systems (Van der Veer & Felt, 1988), and more experienced users (Payne 1991a, 1991b). Researchers also tend to give users more time and repeated practice with the system (Marchionini, 1989; Gray 1990). The numbers of users investigated per study has improved slightly.

The authors of the later studies display a greater scepticism about the suitability of performance data as sole indicators for users' models, both explicitly and implicitly. Payne et al. (1990) emphatically state that successful user performance cannot be taken as an indicator of appropriate users' models: users often complete tasks successfully whilst making verbal statements which indicate inappropriate models. The fact that verbal data are now used in almost all studies (see Table 7) should therefore be seen as a welcome development in the research methods used to investigate users' models. At the same time, the use of verbal data raises a whole new set of methodological questions.

Borgman (1986) raised the possibility of a *model articulation problem*, i.e. that users cannot or do not articulate their users' models. Whether users can give verbal accounts of their users' models is pivotal methodological problem which is discussed in detail in the following chapter (see 5.2.3). The evidence from the empirical studies can be summarised as follows: Payne (1991b) found that experienced users were not able to describe actions verbally, but did perform the same actions without problems when interacting with the system. This can be explained by users' reliance on the display to guide their actions. The *model articulation problem* may therefore apply to verbal data collected outside user-system interaction. However, Van der Veer & Felt (1988), Van der Veer & Wijk (1991) and Payne (1991a) were able to identify users' models in verbal statements collected outside user-system interaction. There are several possible explanations for this conflicting evidence: (a) some users may construct models but others do not, (b)

users construct models of some systems but not of others, (3) some researchers detect users' models in verbal data but others do not.

Authors offer no clear description of how models are inferred from the verbal data or protocols collected in the studies. Many authors do not give any indication how verbal protocols were analysed, and none provide explicit criteria for deciding what constitutes a user's model. Readers can try and infer criteria from the examples of users' models which are provided by some studies (Van der Veer & Felt, 1988; Gray, 1990; Van der Veer & Wijk, 1991); but the examples may have been selected because they corresponded to the researcher's expectations of users' models (i.e. they are UCs which correspond to R(UC)s). The paper by Marchionini (1989) provides a very detailed description of the analysis procedure; it is stated that user's models were inferred from

" ... kind and number of information cited; query formulations, information known about the topic, and information expected through searching on the topic." (p. 599)

Whilst this provides a detailed description of what researchers were looking for in the data, it does not state what constitutes a users' model, and how one decides whether a users' model is appropriate. The collective shyness to publicise criteria for the detection and classification of users' models means that a discussion and integration of results derived from verbal protocols in those studies is not possible. We have to take authors' statements about users' models on trust; they are interesting observations but cannot be treated as established fact. Clearly, this is a methodological problem which has to be overcome to make progress in research on users' models.

Apart from the introduction of verbal protocols, there have been few additions to the research methods employed by researchers of user's models. The most notable additions is Gray's (1990) method of eliciting drawings from users to illustrate their users' model of a hypertext system. Since she collected several drawings in the course of 10 search tasks, this study is also a rare example of an attempt to try and capture the evolving – i.e. dynamic – nature of users' models.

Another methodological problem which emerges from the analysis of the empirical studies is the extent to which users' lack of understanding of real-world tasks may

interfere with construction of the user's models. At least three studies (Borgman, 1986; Gray, 1990; Marchionini, 1989) report evidence that a significant number of users did not have a full understanding of the real-world task. Many of the theoretical contributions see the model of user's task (UT) as the main influence on the construction of the user's model (UC) (see Table 10); thus, if a user does not have an appropriate model of the real-world task (U(WT)), we would expect some difficulty in constructing an appropriate model of the system task (U(CT)). The researchers assumed a level of task knowledge which users did not have. The much-quoted HCI creed that users are experts at real-world tasks (Carroll et al., 1987/88) may be in part to blame. With the benefit of hindsight, Borgman (1986) suggests that users' familiarity or competence with the real-world task has to be examined at the beginning of a study.

4.4.4 Implications for further research

The purpose of the extensive stocktaking and analysis of existing knowledge HCI on users' models conducted in this chapter was to define the starting point and aims of the research undertaking described in this thesis. The main conclusion from this exercise has to be that, whilst there has been considerable research activity on this topic for more than 10 years, existing knowledge is fragmented and far from being integrated into a set of applicable knowledge. The purpose of this concluding section is to outline a research agenda to change this state of affairs and start the development of an integrated and applicable body of knowledge on users' models. The general research agenda can be divided into four main areas:

- unifying terminology;
- integrating existing general knowledge;
- building HCI theories of users' models;
- developing HCI methods for investigating users' models.

The research issues for each area are outlined in brief below. The empirical work conducted as part of this thesis aims to contribute to the final two areas.

Terminology

The state of HCI terminology for users' models reflects the fragmented state of existing knowledge and the diverse research approaches. This state of affairs is much bemoaned (e.g. Carroll & Olson, 1988; Payne, 1992), but the only attempt to break the vicious circle by introducing a framework and unambiguous notation (Nielsen, 1990) has not been taken up by the HCI community. The application of the notation in this chapter has demonstrated that unambiguous labelling of the various types of models is possible. The generic models proposed in theoretical contributions can be grouped into categories and sub-categorise – a possible classification by base models is provided in Appendix 1. The dimensions suggested by Nielsen could be used by researchers to provide profiles of models investigated in empirical studies (see Table 9), which would make comparison and integration of different empirical findings much easier. Finally, the notation has been extended to express the dynamic aspects of model construction; thus, theories of users' models can be expressed as well as individual model types (see Table 10). Since types, categories, dimensions, and processes form the building blocks for the construction of a grounded theory, it turns out that Nielsen's (1990) framework and notation can contribute more than a set of unambiguous labels to resolve the terminology problem: it provides tools which could be used for the construction of a grounded theory of users' models.

Integration of existing general knowledge

The survey of the HCI literature on users' models indicates that little effort has been made to integrate existing general knowledge on mental representations and mental models. Manktelow & Jones' (1987) contribution (see Appendix 2) is the only explicit attempt to translate Johnson-Laird's (1983) theory of mental models into design principles, and it has been all but ignored. Despite the widespread adoption of the term and the notion of mental models in HCI, it is only recently that the application of Johnson-Laird's (1983) theory to users' models has been considered again by Payne (1991a, 1992). In 3.4, 3 reasons were identified why the theory of mental models might not be applicable to users' models of computer systems, all of them rooted in the fact that the task of interacting with computers has additional cognitive dimensions to those tasks which provide the empirical foundation for

Johnson-Laird's (1983) theory. The review of the HCI literature on this topic confirms the validity of that concern. Interacting with computers not only adds a cognitive dimension; interacting with computers involves more than cognition. Users rely on visual cues and motor feedback (Mayes et al., 1988; Payne 1991b). Furthermore, cognitive performance is affected by other factors such as fatigue (Olson & Olson, 1990). Despite these reservations, the mechanisms which Manktelow & Jones (1987) suggested apply to users' models emerge from the results of the empirical studies. Users seem to employ existing models as an *advance organiser* of incoming information, there is evidence of *linguistic cueing*, and the emphasis on users' tasks reflects the notion of *content-dependency*. Mental models theory would have predicted the difficulties untrained users have with abstract notations such as Boolean algebra (Borgman, 1986; Marchionini, 1989). Similarly, the breakdown of metaphors which try to exploit surface similarities, but provide no explanation of the internal working of a system, ties in with Johnson-Laird's (1983) theory. Clearly, the application of the theory to the cognitive mechanism of users' models deserves further consideration. Manktelow & Jones' (1987) Summary Design Principles provide a starting point for empirical investigation, but the principles would have to be reformulated and exemplified to make them accessible to designers.

There is little discussion concerning the use of representations in computer systems, and their relationship to users' models. Claims that icons are easier to use than text labels (Apple Computer Inc., 1987) seem to be based on the (incorrect) assumption that all pictorial representations have picture-like qualities and are therefore easily recognised. It is feasible to represent a specific user's task through an isomorphic mapping by designing a dedicated system image which mimics the physical task and represents individual objects and tools - many direct manipulation interfaces are designed following this approach. However, representing the general capability of a computer – which is an abstract machine manipulating symbols – through picture-like representation poses an interesting problem. As Payne (1992) points out, computers employ representations of themselves, and modify these internal representations. There is a potential conflict between *simplicity* (represent only what is needed) and *transparency* (reflect the internal working of the system) on which

guidance should be given to designers. The relationship between representations and users' models warrants further research.

Attempts to apply analogical reasoning or metaphorical reasoning to users' models have been restricted to instructional studies which tried to induce such a mechanism via instructional design models (DCs). The attempts have met with limited success; certainly, most users' did not reproduce the DC as part of their user's model to the extent in which they have revealed source models and parallel inference mechanisms in studies of natural phenomena (Gentner & Stevens, 1983; Collins & Gentner, 1987). Wozny (1989) suggests that users need some existing knowledge of computers to apply analogical reasoning, since models from outside the computing domain do not provide sufficient structural similarities. As outlined in 4.2.5, structural similarity is linguistic. Therefore, mapping to an existing task or tool could work if it is supported linguistically by the system image. Thus, attempts to explain a new system in terms of an existing device (*"a word processor is like a typewriter"*), without supporting that model in the system image, are not likely to succeed.

Building HCI theories of users' models

Theoretical and empirical work on users' models have been conducted without much reference to each other. The exception is Stephen Payne's work – which is considered further on in this section. The remaining theoretical contributions therefore remain largely at the level of untested conjectures.

One reason for this state of affairs may be that putting a theory such as conceptual design to the test would be an ambitious and expensive undertaking, since it involves designing and implementing a system or application. One could envisage such a study being undertaken in the context of a design project: this type of research would require long-term commitment and a high degree of involvement. Most academic institutions would lack the resources for such a project. Conceptual design would seem to be an ideal candidate for an industry-based HCI research project along the lines advocated by Carroll & Campbell (1986, 1989) and Monk & Wright (1990), but no such project has been reported in the literature to date. The basic idea of conceptual design – creating a system which supports the construction of appropriate users' models of itself – undoubtedly appeals to both HCI researchers

and practitioners, as its inclusion in most HCI textbooks (e.g. Preece et al., 1994; Newman & Lamming, 1995) and practitioner's guides (e.g. Tognazzini, 1991) demonstrates. The problem is the lack of "how-to" knowledge required to put the idea into practise. *"What does a user's model look like? How do I go about constructing a design model? What is the difference between a user's model which works and one which does not? Can you develop a design model when you do not have access to the users?"* These are typical questions raised by student-designers and practising designers about the conceptual design approach; in its existing form, conceptual design is simply not precise enough to be applied as part of a design process.

The current state of fragmented knowledge can only be overcome by building more precise theories of users' models. Newell & Card (1985) regarded mental models as a candidate for an engineering-style theory; the review reveals, however, that no such HCI theory can be formulated at present. The relationship between users' models and performance, which would be crucial to the formulation of both an engineering-style theory and engineering principles (Dowell & Long, 1989) is anything but clear. The relationship between users' models and tasks, another foundation required for an engineering-style theory, needs to be specified in more detail. Most theoretical contributions suggest a connection between tasks and design models (DC) and/or tasks and users' models (UC), but with the exception of YSS (Payne et al., 1990), this connection remains an intuition which is expressed at the level of craft knowledge (e.g. *"user's task should be taken into account."*) YSS, labelled as a hypothesis by its authors, is the most precise theory put forward so far, and the only one to have been subjected to empirical investigation. Even YSS, however, cannot offer hope of an engineering-style theory at present. Payne (1991b) indicates that YSS is not sufficient to explain user performance:

"To understand action at the user interface, it seems models will be needed of the way that actions can be performed in the absence of perfect knowledge of operator effects. Such models will be needed to stress the way information from the display is interpreted during and in support of actions sequences." p. 288

Obviously, YSS could be refined, and further experiments could be conducted to demonstrate that it does not only apply to the simple text editing tasks with which it has been evaluated so far. But from the point of view of generating a body of

knowledge of users' models, putting all effort towards a traditional science approach to develop and "prove" one hypothesis such as YSS would be high-risk strategy.

A more prudent and practical approach, which does not exclude other existing knowledge, is to build a theory of users' models from existing knowledge and examples. We do not have a clear understanding of the phenomenon at the moment, hence, we should obtain precise descriptions (see 2.3). The lack of descriptions of users' models and design models is one of the reasons why most of the theoretical contributions are lacking precision and applicability. Only three of the of the empirical studies (Van der Veer & Felt, 1988; Gray, 1990; Van der Veer & Wijk, 1991) publish examples of users' models elicited. These examples come in a range of formats (sketches, drawings, if-then rules, programming languages, etc.) – basically, any description of the system straight from the user's mouth or pen. Given the paucity of examples of users' models, and the diverse nature of the few examples which have been published, sceptics may argue there is no evidence that users' models actually exist[32]. Thus, it is not surprising to find that the first two points on the list of 11 basic research needs drawn up by Carroll & Olson (1988) are:

(1) provide a description of what a users' model looks like; and

(2) provide evidence that users actually have models and use them when interacting with the system.

It may seem strange that such basic evidence needs to be acquired after more than 10 years of research into this topic, but the results from the literature survey show the consequences of failure to address this requirement earlier. The reason for underlying this neglect is revealed by Card & Moran (1986):

"Although we were very concerned about the mental model issue, we didn't pursue explicit studies for several reasons: we didn't have a satisfactory methodology for studying it, we didn't have satisfactory representations of it, and we were busy pursuing the performance issues [...]". p. 191

---

[32] This point of view is not represented in the literature, but occasionally raised in discussions at conferences.

The lack of HCI methods for obtaining evidence and descriptions of users' models is a methodological problem which needs to be resolved first.

Developing HCI methods for investigating users' models

The review of the empirical work on users' models to date has revealed that user performance cannot be taken as a reliable indicator of users' models. Given that HCI has not produced suitable alternative indicators, we may have to follow Carroll & Campbell's (1986) advice and collect *"all the data we can get"* in studies on users' models. Verbal data and verbal protocols are collected with most studies now, but the use of such protocols raises methodological problems of its own. How are users' models to be detected in verbal protocols? Are such protocols reliable indicators of the models which users hold? Attempts to encourage users to verbalise their thoughts for inspection by the researcher may interfere with users' models or the construction process. These problems are discussed in detail in the following chapter, which also introduces the set of studies designed and conducted as part of the empirical work of this thesis.

5 Investigating users' models

The aim of the empirical work described in this thesis was to address two of Carroll & Olson's (1988) basic research needs on users' models:

(1) to procure evidence that users' models actually exist, and

(2) to detail what users' models actually look like.

These research needs were addressed by conducting a series of exploratory studies which observed users interacting with computer systems, capturing users' behaviour and their interaction with the system in as much detail as possible. If the evidence for users' models could be found in and descriptions generated from the data, such a detailed set of data could also be used to evaluate existing theoretical statements about users' models, and thereby serve as a basis for formulating HCI-specific theories of users' models.

In the planning of the studies, two methodological questions had to be addressed first:

(1) which investigation scenarios are suitable for studying users' models; and

(2) which methods of data collection and analysis are most likely to detect evidence of users' models.

This chapter presents a detailed discussion of the issues pertaining to these questions, and explains how the conclusions led to the design of a set of scenarios and methods which were applied in 5 observational studies on users' models.

5.1 Design of the studies

When designing empirical studies of users' models, HCI researchers have to trade off internal against external validity. *Internal validity*, which is highly prized in traditional psychology experiments, can be achieved by eliminating or controlling any influences on the phenomenon under investigation. Efforts to ensure internal validity tend to result in experimental setups which are highly artificial, and the results obtained under such conditions may not describe the behaviour of the phenomenon in the real world. In observational studies without experimental

variation, internal validity means that we try to ensure that the different observations of the phenomenon take place under similar circumstances, so that observations can be compared with each other.

External validity can be maximised by studying the phenomenon *in situ* to ensure that all conditions which affect it are investigated. In HCI, this would usually require that we observe users in the workplace. Such studies have been rare in the past, because it was difficult to obtain access to a significant number of users in any one place, and implement unobtrusive means for capturing data. The situation has changed somewhat since the introduction of ethnomethodology to HCI problems by Suchman (1987), at about the same time as the observational studies reported in this thesis were designed.

The aim in the design of the studies was to find a balance of internal and external validity which suited the phenomenon under investigation. These considerations applied to the choice of users, tasks, and system observed. The researcher can make choices for each of these; the choice should be driven by how conducive it is to provide observations of the phenomenon – users' models. After a brief discussion of the likely impact of characteristics of users, tasks, system and environment, conclusions are drawn for the design of the observational studies.

5.1.1 Users

From the review of the theoretical contributions in Chapter 4, we know that users' existing knowledge and experience is likely to influence the users' models they construct. Differences may exist in users'

(1) general knowledge and experience (UW);

(2) knowledge and experience of the particular system investigated (U(CT));

(3) knowledge and experience of the real-world task (U(WT)).

Significant differences in any of these are likely to impact on users' models. A large heterogeneous user group would be best to ensure external validity of the results, but such studies require many resources. Smaller heterogeneous groups may yield very diverse observations; this could make detection of patterns in the data very

difficult. It is easier to detect similar users' models and problems in homogenous user groups.

Many of the theoretical contributions suggest that users' model are highly individual, since they depend on users' general knowledge and experience (Manktelow & Jones, 1987). Putting together user groups with similar general knowledge and experience may therefore seem an impossible task. In the past, studies have tried to address this problem by recruiting users with similar backgrounds (students on a course or employees in a particular company).

As far as knowledge about the computer system used in the observation is concerned, it is easiest to ensure a similar level of knowledge if that is level is "none". This has led to a the predominance of studies involving first-time users. Many studies have focused on the early stages of user-system interaction, when users' model may still be "under construction" and not sufficiently well established. This not only reduces the external validity of the results, but may also affect users' confidence (*viz.* the study by Payne et al., 1990) and their willingness to verbalise their thoughts. Observing very experienced users, on the other hand, may yield good performance, but few data which provide insights into their users' models (see 5.2.2).

The review of the empirical studies (see 4.4.3) concluded that in the past, researchers have not always paid sufficient attention to users' knowledge of the real-world task (U(WT)). Differences in task knowledge, or insufficient task knowledge, can have a major impact on users' models. Users' task knowledge can be checked (e.g. through benchmark tests); alternatively, the investigator can select tasks which are part of users' general knowledge and experience.

5.1.2 Tasks and task-context

Given the close link between users' models (UC) and users' tasks (UT), to achieve internal validity, users ought to work on the same tasks, possibly even in the same order. This can be achieved by structuring user-system interaction tightly, i.e. decomposing the real-world tasks (U(WT)) into sub-tasks which are related to particular system functions. Tightly structured interaction, however, may deprive users of the meaningful task-context which Payne (1992) advocates to ensure

external validity of the observations. Another drawback of tightly controlled experimental scenarios is to raise users' awareness of "being observed", and therefore may change the *demand structure* of the interaction situation (see 5.2.3). Even when user-system interaction is tightly structured, users will create different situations in the course of the interaction (e.g. in their attempts to correct errors) and different interactions and responses will be required from the researcher.

These are concerns relating to the real-world tasks (U(WT)) which the system would be typically applied to. In the design of the studies, one of the main concerns was to create a situation and task context which was conducive to eliciting users' models. This is a point hardly ever discussed in the literature. One of the few suggestions made is that by Young (1983), who proposes that to find evidence of users' models, researchers should observe:

(1) users *using* the system;

(2) users *explaining* the system;

(3) users *predicting* the behaviour of the system;

(4) users *learning* the system.

Empirical studies of users' models to date have put the first and fourth suggestions into practise, whereas the use of explanation and prediction to elicit users' models has not been tried. Explaining a computer system and predicting its behaviour should put users' models to the test (DiSessa, 1986); they are also activities which have a natural affinity with verbal statements. *Thinking aloud* is highly artificial behaviour (see discussion in 5.2.3); scenarios which require dialogue are less artificial because verbalisation of users' thoughts is necessary part of a higher-level task. A *constructive interaction* scenario (Miyake, 1984) was successfully employed in the study by Van der Veer & Wijk (1991); such scenarios can be expected to yield a considerable amount of verbal data. The drawback is that it can be difficult to structure user-system interaction in such studies, since users determine the direction of activity.

5.1.3 System

The review of empirical studies in 4.3 revealed that all recent studies investigated users' interaction with, or users' models of, commercial computer systems and software. The use of real systems, rather than artificial devices created for especially for studies, helps to increase the external validity of the observations.

Since users depend on the information flow from the system (Payne 1991b), data should be collected during user-system interaction rather than outside it. Payne (1992) emphasises that HCI needs to depart from the investigative methods of tcognitve psychology to investigate users' models:

> *"There seems to exist an unspoken argument underlying traditional approaches to cognitive skill: only by looking at unaided skill can we see the mind in action." p. 105*

5.1.4 Conclusions for the studies

Users

To obtain a set of comparable observations, it seemed advisable to use a homogeneous user group with a similar background. The group recruited for the study was a group of art students taking a subsidiary course (Introduction to Computing). None of the users had any knowledge or experience of the systems they were interacting with; the difference to previous studies of novice users was that the 4 of the 5 observations did not take place during the early stages of learning, but after a several weeks of learning and practice (see Table 11). This should have given users sufficient time to construct and consolidate their users' models of the systems to which they were introduced. An exception was the final study, in which users were observed learning a second word processing system; the purpose of this study was in investigate transfer of users' models between two different systems used for the same task.

Rather than conduct a series of one-off observations, it was decided to observe the same users interacting with a range of systems, since such a set of observations would allow comparisons of individual users' models of different systems over time. Such a comparison could also reveal dynamic aspects of users' models (changes over

time), and idiosyncrasies in individual users' models which appear in a users' model of different systems. It was therefore decided to investigate a group of users who learned to use a range of computer systems over a period of 6 months.

The study also included two comparison groups with different levels of background knowledge and experience than the users in the group. The users in the comparison group for the spreadsheet study had a background in mathematics and programming. The comparison group for the second word processing study had a near identical background to the users in the main group, but were complete computer novices at the time of the observation. The purpose of running these control groups was to see if different levels of relevant background knowledge would manifest themselves in differences in user behaviour and users' models.

Tasks and task-context

A series of 5 observational studies was devised with the aim of eliciting users' models from a cohort of users described above. The tasks presented to users in the observational studies were mainly real-world tasks around which instructions and practical work during their course had been centred (see 5.4.1). Users were asked to edit a letter on the word processor systems, set up a spreadsheet on the spreadsheet system, run queries on the database, and amend a Prolog program on family relations.

Studies 1, 3 and 4 included transfer tasks, i.e. tasks requiring user actions which the users had not practised previously, but could be inferred from the information presented to the users through instruction. In these studies, the author interacted with users to encourage them to provide verbal accounts of their reasoning during user-system interaction. Encouraging users to think aloud in this manner may change the *demand structure* of the interaction situation, and thus the phenomenon under investigation (see discussion under 5.2.3).

In an attempt to make the investigators' encouragement of verbalisation less obtrusive, a different approach was devised for studies 2 (see Chapter 7) and 5 (see Chapter 10). These studies employed co-investigators, presented as novice users or fellow learners, who would engage users in dialogues about their actions and system behaviour. Since these two studies promised to provide rich data, comparison

groups were recruited to see if different models might be elicited from users who had the same level of expertise as far as the computer system was concerned, but brought a different type of background knowledge to that system (see 5.3.4)[33].

Course week	Instruction period (weeks)	Computer system	Method	Scenario
5	4	WordStar	using system	tasks (hands on)
9	4	Spreadsheet	explaining system	teaching back
15	5	Prolog	predicting system behaviour	tasks (verbal and hands on)
19	3	Database	knowledge and using system	tasks (verbal and hands on)
21	none	MacWrite	learning system	constructive interaction

Table 11: Overview of observational studies

System and environment

Users were due to be introduced to a range of commercial software systems as part of a course they were taking: a word processor, a spreadsheet, a database and a programming language. User interaction with these system was observed in the studies. These systems, with the exception of Prolog programming, are typically used in office environments. Therefore, an office-like environment was created in the laboratory in which the sessions were conducted and recorded. Laboratory environments and the use of video cameras can be intimidating. To counteract this, a small laboratory was set up which contained an office-type setting (a computer on desk, surrounded by bookshelves).

5.2 Detecting users' models in observable data

The discussion of empirical studies of users' models in the previous chapter (see 4.4.3) concluded that researchers cannot assume a particular users' model from user

[33] Ideally, such comparison groups would have been run for all studies, but limited resources (time) meant this was not possible.

performance data alone. This raised the question of which data should be collected by researchers trying to collect evidence of users' model. Carroll & Olson (1988) suggest that new methods of data collection and analysis may have to be devised. This section provides a detailed discussion of the 3 types of data typically collected (performance data, on-line protocols, and verbal protocols) and describes the method of data collection employed in the studies.

5.2.1 Performance data

Performance data have been employed in virtually all empirical studies of users' models to date. These are typically collected through benchmark tests (usually conducted after training sessions), which assess users' performance on a set of tasks, often including a transfer task. User performance is typically assessed in terms of:

- task completion;
- time taken to complete tasks;
- number of errors;
- retention over time.

In the conclusions of the methods section of empirical studies on users' models (see 4.4.2), problems with the use of performance data as indicators for users' models have been raised. Performance cannot be taken as an indicator of a specific users' model (UC), i.e. a model of a particular content and structure. The studies of instructional design models (DCs) demonstrate that assumptions that users who perform well have adopted a particular model cannot be substantiated. As Carroll & Rosson (1987) put it, users' minds are not blank slates onto which externalised models can be printed. A related but slightly different point raised is that good performance from a user cannot not be taken as proof of a model which a researcher would regard as an appropriate model of the system; as Payne et al. (1990) put it, users who perform well can harbour the most appalling misconceptions about the system.

Even though performance data may have been over-interpreted in early research on users' models, they should not be discounted as indicators of users' models. In general, difficulties in completing a task do indicate lack of knowledge or failure to

apply it. Manktelow & Jones (1987) point out that incorrect user's models should lead to systematic rather than random errors. Recorded for detailed analysis, user problems with task completion can help the researcher to identify parts of on-line and verbal protocols in which such errors may be evident, and further analysis may yield insight into users' models.

5.2.2 On-line protocols

Norman (1983) advocates the use of on-line protocols which record user and system actions. Since they can be collected in a completely unobtrusive manner, he considers them to be more reliable than verbal protocols. Given the right equipment, on-line protocols are easy to collect and analyse, providing an instant summary of frequency of use, errors, etc. On-line protocols are therefore widely employed in usability labs for purposes of system evaluation. They have an advantage over performance data in that they allow the researcher to link user actions to system actions, and thus allow the researcher to reconstruct the flow of information from the system display to the user at the time of a particular action. This information is a major requirement for explaining user action (Payne, 1991b), but it may not be sufficient: a researcher's conceptualisation of a users' model (R(UC)) derived from on-line protocols only is bound to reflect a certain amount of guesswork. A richer record be obtained by integrating relevant user comments (obtained from verbal protocols) into the record of user-system interaction (see 5.2.4).

5.2.3 Verbal data and protocols

Verbal data and protocols are now collected in most studies of users' models. The discussion in 4.2.2 concluded that verbal data obtained in isolation from the system are not likely to be an accurate reflection of user competence, given users' dependence on the information flow from the display (see above). Verbal protocols collected during user-system interaction ought to provide a better reflection of users' models and the ways in which they are used. But there are a number of problems connected with the use of verbal protocols to capture users' models.

At the root of these problems is the well-known psychological fact that people cannot give complete verbal accounts of their thought processes (Norman, 1983). The construction of a user's model is mainly a subconscious process (particularly where encoding and processing of propositional representations are concerned), and subconscious processes are not verbalised. A researcher can, of course, ask users to give a verbal account of their reasoning, but a researcher who treats these accounts as true reflections of users' decision processes would be falling for what Johnson-Laird (1983) describes as the *fallacy of conscious access*. People are not aware of the processes involved in their reasoning, only of the products or results of these processes. Thus, users' verbal accounts should not be interpreted as accurate reflections of users' decision processes, but as an evaluation of their decision outcomes (Manktelow & Jones, 1987).

A related problem is that, when a researcher asks users to give a verbal account of their reasoning, the shift from subconscious to conscious processing may change the nature and content of the reasoning process. The researcher changes what Norman (1983) calls the *demand structure* of the interaction situation; users' responses may be influenced by what they think is required of them. Users may:

- provide post-hoc rationalisations of their actions;
- say what they think the researcher wants to hear;
- be guided by subtle verbal and non-verbal cues given by the investigator.

Thus, verbal protocols can actually produce misleading information.

Despite all these reservations, verbal protocols offer a rich source of data from which users' models could be reconstructed. They may not provide a record of users' reasoning processes, but the outcome of the reasoning processes will reveal some indication of the elements and mechanisms involved in them. Outcomes of reasoning processes are the closest researchers are likely to get to the processes themselves, especially if they can be tied to user actions and system display. As for the effects of the demand structure on users' statements, Payne (1991a) demonstrated that verbal cues can help to distinguish existing users' models from those which are generated in response to questions. Researchers should also

consider that tightly controlled experimental scenarios are more likely to cause a change in demand structure than loosely controlled ones (see 5.1.2).

5.2.4 Conclusions for the studies

> *"Let me warn the nonpsychologists that discovering what a person's mental model is like is not easily accomplished." (p. 11)*

The preceding discussion confirms the prophetic value of Norman's (1983) footnote: by themselves, neither performance data, on-line protocols nor verbal protocols may provide reliable evidence of users' models. Since each of them can provide some evidence, the author devised a new method for capturing and combining these three types of data. Given the aim and exploratory nature of the studies, collecting *"all the data we can get"* (Carroll & Campbell, 1986) is the appropriate strategy.

Traditionally, audio recordings have been used to obtain verbal protocols. In previous work with such protocols (Sasse, 1986) the author had encountered problems in linking users' verbal accounts to system states. It is possible to relate verbal and on-line protocols through the use of time-stamps, but the method is cumbersome in both the capture and analysis phases. The author devised a method of combining on-line and verbal data on a single medium. The screen was recorded from the system unit via a mediator box. The user and (co-)investigator were recorded with a camera – all actions performed by the user, changes in display, and users' verbal and other behaviour (including interaction with the (co-)investigator) could thus be captured. The images were combined using the overlay function on a video mixer, and recorded with the sound onto a single videotape. The resulting recordings provided a simultaneous access to

- the computer screen (providing an on-line protocol of user-system interaction);
- a record of user behaviour; and
- verbal protocols.

The recordings provided a very rich protocol of the sessions; it is possible not only to work out what users could see at particular point in time, but whether they were looking at the display, where they were focusing, pointing etc. when making certain

statements. The investigative scenarios were designed to encourage verbal accounts from users, whilst minimising the change in demand structure (see 5.1.2).

5.3 The studies

5.3.1 Users

Users were students at the University of Birmingham, taking a subsidiary course in computing. This was an introductory course, advertised for students with no previous computing experience, and open to undergraduate students from the arts faculty. The course ran for two terms (10 weeks in each term), and students attended for three hours per week: a one hour lecture on the theory of computing[34] and two hours of tutorials.

These tutorials were planned and conducted by the author, and consisted of instructions and hands-on exercises introducing a range of application software. The tutorials and instruction materials were designed by the author following the minimalist training model (Carroll et al., 1987/88), which has four main principles:

(1) *Focus on real tasks and activities*: Most people who learn an application system are doing so because they believe a that the system is a tool which might help them to do their work. Training should capitalise on this motivation, and draw on the learner's task-relevant knowledge, by introducing functionality in the context of recognisable tasks. This principle was implemented by organising tutorials and coursework around high-level tasks which the students could relate to, such as writing letters (word processing), working out students' marks (spreadsheet), producing mailshots and chasing bad debts (database), and writing a database and rules for a restaurant booking agency (Prolog programming).

(2) *Slash the verbiage:* Most people who learn an application system do not want to read through vast amounts of complex information, and feel confused and intimidated by computing jargon. Learners should not be

[34] These were given by Peter Greenfield, a lecturer in the School of Computer Science.

swamped with instruction material, nor be expected to be familiar with computing terms. This principle was implemented by writing concise instructions rather than copying manual pages, and by explaining basic terms (such as editing and copying) and demonstrating procedures (such as inserting a diskette).

(3) *Support error recognition and recovery:* Most people who learn an application system will make mistakes during their learning process. Training should capitalise on this by teaching learners how to diagnose errors and how to recover from them. Implementation of this principle was achieved by incorporating the following exercise in the tutorials: one student would sit at a machine that was linked to a television. The tutor would ask him or her to perform a task (which offered the opportunity to make errors) while "thinking aloud". When the student made a mistake, or could not work out what to do next, the tutor would start a dialogue with the student about what caused the error and how they could recover from it. The rest of the group watched and were encouraged offer help and explanations.

(4) *Guide Exploration:* Guided exploration incorporates principles 1 and 3, and is essentially *learning-by-doing* – encouraging learners to discover the system and its functionality rather than introducing functionality in systematic, step-by-step instructions. It was attempted to implement this principle by allowing students to choose their own coursework after they had been introduced to the basics of a software package. For their coursework, students were encouraged to do a task they had to do for their course or other activities, e.g. type up a report or an essay (word-processor), do the accounts for a student society event (spreadsheet), or create an inventory or address book (database).

5.4.2 Recruitment procedure

There were a total of 32 students on the course. All students received a handout from the author at the beginning of the course, informing them that the course tutor was conducting a series of experimental sessions to evaluate usability of the software

packages and the way in which they had been introduced to the students. The students were asked to participate in the series of 5 studies, but it was emphasised that participation was entirely voluntary and would have no effect on the student's course marks. The observations took place after the students had used the computer systems for 3-5 weeks (apart from Study 5).

			WP	Spread-sheet	Prolog	Data-base	Mac-Write	total
Subjects	Sex	Age						
1	male	38	y	y	y	y	y	5
2	male	19	y	y	y	-	y	4
3	female	19	y	y	y	y	y	5
4	male	18	y	y	y	y	y	5
5	male	19	-	y	y	y	y	4
6	male	19	-	y	y	y	y	4
7	male	19	y	y	y	y	y	5
8	female	20	y	y	-	y	y	4
9	male	18	y	y	y	y	y	5
10	female	20	y	y	y	y	y	5
11	female	19	y	y	y	y	-	4
12	female	20	y	y	y	y	y	5
13	female	19	y	y	y	y	y	5
14	female	20	y	y	y	y	y	5
15	male	18	-	y	y	y	y	4
16	female	20	y	y	y	y	-	4
17	female	18	y	y	y	-	y	4
18	female	19	y	y	-	y	y	4
19	female	20	-	y	y	y	-	3
20	female	18	y	y	y	y	y	5
total			16	20	18	18	17	

Table 12: Subjects participating in studies

Subjects in the main group

Out of the total number of 32, between 21 and 26 students attended and completed the experimental sessions. All sessions were transcribed, but it was decided to remove all subjects who did not participate in at least 3 of the 5 studies. This left a group of 20 subjects, 12 female and 8 male, aged 18-20, except for one mature student who was 38 (see Table 12). They were attending courses in geography, music, politics, physical education, languages or general honours. A short interview

at the beginning of the course with each of them (see Appendix 2) showed that, even though most of them had actually touched a computer once before they came on the course, none had significant experience in programming or using application software on a PC. All seemed motivated to learn about computers and get hands-on experience. Many of them felt that they had "missed out" on computing at school, and thought that they would have to use computers during their course or in subsequent jobs. No payment was offered to this group of students.

Subjects in the comparison groups

Two additional groups of subjects were recruited to explore the constructive interaction scenarios with users who brought different background knowledge to those systems. For Study 2 (spreadsheet), 20 subjects were recruited from a group of first year Software Engineering and Maths with Computing undergraduate students who had learned to use the same spreadsheet as the experimental group. For Study 5 (MacWrite), 16 subjects were recruited from the subsidiary course at the beginning of the following academic year, and asked to explore the system together with another user (co-investigator) before their first course tutorial. All subjects in the comparison groups were paid £3 for participating in the one hour sessions.

5.4.3 General procedure

Observation sessions consisted of users being asked to work through a set of questions and/or hands-on tasks. In studies 1, 3 and 4, an investigator (the author) would sit with the users as they worked through the tasks, prompting them for verbal information whenever appropriate. The investigator would always ask for explanation when errors occurred, and when users seemed not sure what to do next. Users could ask questions – the investigator would usually ask counter-questions to probe users' models. Only when users seemed unable to solve a task or recover from a problem would the investigator give them procedural instructions (see also the detailed description of the studies). In the studies which employed constructive interaction scenarios (2 and 5), users would be introduced to a learner or co-learner, who really was a co-investigator trained by the author to probe users' models. The video mixer and recorder were hidden behind a screen. Users were able to see the

camera which recorded them and the co-investigator, but it was out of their line of view when they were looking at the computer screen or at the (co-)investigator.

All observation sessions were recorded on videotape (see 5.4.2) and subsequently transcribed. Comments and explanations have been added to users' verbal statements and system actions by the author where appropriate. For the summary tables presented in chapters 5-10, tasks were scored as being successfully completed – or not – by the author on the basis of the videotape rather than transcripts. When marking the tasks as completed or not completed, the following rules were applied throughout:

(1) A task would be marked as completed if the user could solve it without any prompting or help from the investigator or co-investigator.

(2) If the user made an error, realised an error had occurred, and corrected it without any help or prompting from the investigator or co-investigator, the task would also be marked as completed.

(3) If the user needed any help or prompting to complete a task or recover from an error, the task would be marked as not completed.

The detailed analysis and discussion concentrated on tasks which most users could not complete successfully. As discussed under 5.2.1, systematic errors may be an indication of an incorrect users' model. In addition, it was found that non-completed tasks often resulted in dialogues between users and (co-)investigator, and therefore yielded verbal responses which might contain indication of the models users were trying to apply.

The following five chapters give a detailed description of conduct and results of the five observational studies.

6 Word processing study

The purpose of this study was to elicit users' models of a word processing system. 16 users were observed using the word processor after 4 weeks of training and practise. The task scenario was tightly structured, with users working through a sequence of 11 tasks; the investigator would encourage users to think aloud. At the end of the session, they were also asked to describe the system to the investigator.

Task	Description
1 Copy file	Copy the file called STUDY.DOC into a new file.
2 Load file	Edit the copy of the file.
3 Del char insert word	Delete a character in the first line of the document, insert a word
4 Insert paragraph	Insert a paragraph (subject's address) in the top right hand corner), as you would in a letter.
5 Move paragraph	Move paragraph from one part of the document to another.
6 Delete paragraph	Delete a paragraph
7 Copy para – file (TT)	Copy paragraph containing address into a new file.
8 Include file (TT)	Include an existing file (INCLUDE.DOC) into current document.
9 Delete block	as 6
10 Save File	Save the current file and return to the Opening Menu
11 Rename file	Rename edited file
12. Describe a WP	If you were explaining to a friend who has never used a computer before what a word processor is, how would you describe it?

Table 13: Word processing tasks

6.1 Procedure

The scenario of this first study was "observing users using a system", in a set-up which remained close to the typical experimental set-up employed by most of the studies reviewed in 4.3. Users were asked to edit a document on a keyboard-based word processor (WordStar) which they had learnt to use and practised with for four

weeks previously. The document on which users worked in the session contained the instructions for the 11 file operation and editing tasks to be performed (see Table 13) – a somewhat artificial task. All the procedures needed to complete the eleven tasks had been covered in the instruction and practice sessions, except for two functions (*7 - copy paragraph to file* and *8 - include file*), which served as a transfer tasks (TT). Successful completion of a transfer task - which requires users to infer new procedures on the basis of those that have been taught - is regarded by many experimenters (e.g. Kieras & Bovair, 1984; Borgman, 1986) as a test for a successful users' model. At the end of the session, which lasted approximately 30 minutes, each user was asked to describe a word processor in their own words.

The investigator (the author) sat next to the users, who were allowed to ask questions if they did not know how to perform a task. The elicitation strategy in this session was to encourage *thinking aloud* in the same manner as in previous HCI studies. The investigator encouraged users to comment on their actions, and kept prompting them to reason about any mistakes/errors. The investigator applied the following rules throughout the session:

(1) If the user asked the investigator how to solve a certain task, the investigator would not answer by telling the user the procedure in the first instance. Instead, the investigator would ask a counter-question such as: *"How did you delete (insert, underline) a word (sentence, paragraph) in the exercises in class?"* Only when it was clear that the user could not remember the procedure, the investigator would tell the user how to solve the task.

(2) Whenever a user made an error, the investigator would wait to see if the user would comment on the error, try to recover from it, or try again. If the user did not say or do anything following the error, the investigator would try to prompt the user to comment on the error, and how to recover and perform the task. Typical questions used to prompt users were: *"What do you think happened there? Why? What did you expect the system to do? "*

(3) Whenever the user made a comment that indicated s/he was referring to some kind of user's model, the investigator would encourage the subject

to elaborate their statement (e.g. by asking *"What exactly do you mean by that?"*).

6.2 Results

The transcripts of these observation sessions are provided on pages 1-22 of the Transcripts volume, an overview of the results is given in Table 14. 16 users participated in the study, and managed to complete between 4 and 9 of the 11 tasks successfully. More than half of the users encountered problems with tasks

4 insert paragraph (14)

5 move paragraph within document (8)

10 save file (9)

and the transfer tasks

7 copy paragraph to separate file (16)

8 include file in document (12)

The reasons for the problems encountered, and their implications for users' models (UCs), are analysed in detail in the following sections.

6.2.1 Inserting text at the beginning of a document

Analysis of the tapes revealed that all users knew how to insert text, but most failed to apply this knowledge because the text had to be inserted before the first line. Only two users required no assistance to work out that they could use the CARRIAGE RETURN key to insert the required blank line. Typically, users would position the cursor in the first line, press the UP ARROW key, and be very surprised when the cursor started moving to the right. After some prompting (*"Why do you think it won't move up? Where exactly in the document are you at the moment?"*) did users realise that they were at the beginning of the document. Most then realised that they had *to "create some space on top of the file"*, or *"move the text down"*. When asked about the easiest way to achieve this, most users went straight for the CARRIAGE RETURN key. A few users, however, persisted with the scrolling commands, and some suggested to mark the text and move it down.

Once they had inserted a few blank lines, users had to type the text in the right-hand corner. Most tried the RIGHT ARROW key (which moved the cursor down) before inserting blank space with the SPACE BAR or TAB key. Three users could not work this out at all, and the investigator had to explain to them that they had to insert spaces.

6.2.2 Transfer tasks

All users failed to perform the first transfer task (copy paragraph to separate file), and only four managed to perform the second transfer task (include file in document), which was the reverse operation. With the first transfer task, 10 users suggested leaving the document to create a new file. When told that it was a command available in the menu options, most opted for commands which they knew – COPY and MOVE – even though they had used both commands extensively in course exercises for copying and moving text in the same document. Users were not able to guess the WRITE command for the first option. In the second transfer task, most users again suggested leaving the document to start looking for the file. Only 4 users opted for the READ command straight away, *"because it's the same as WRITE in reverse"* (e.g. Subject 7, Transcripts-7). Others tried to use WRITE again, and COPY, and COPY FILE. Many users also suspected the command was called READ because it would allow them to read the file to check it was the right one before it was included.

6.2.3 Move paragraph

This was a command that users were familiar with from the training sessions, but half of them still did not manage to complete the task without prompting or help. All users remembered some parts of the operation, but the most striking observation was that those who needed help had to be reminded that the menus actually contained the information that they were looking for.

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Total y
1 Copy file	y	y	n	y	-	-	y	y	y	y	y	n	y	y	-	y	y	y	-	y	14
2 Load file	n	y	y	y	-	-	y	n	y	y	y	n	y	y	-	y	y	y	-	y	13
3 Del char insert word	y	-	y	y	-	-	y	y	y	y	y	y	y	y	-	y	y	y	-	y	15
4 Insert paragraph	n	n	n	n	-	-	n	n	n	y	n	n	n	n	-	n	y	n	-	n	2
5 Move paragraph	n	y	y	n	-	-	y	y	n	y	y	n	y	n	-	n	n	y	-	n	8
6 Delete paragraph	y	y	y	y	-	-	y	n	n	n	y	y	n	y	-	y	y	y	-	y	12
7 Copy para – file (TT)	n	n	n	n	-	-	n	n	n	n	n	n	n	n	-	n	n	n	-	n	0
8 Include file (TT)	n	n	n	n	-	-	y	n	n	y	n	n	y	n	-	y	n	n	-	n	4
9 Delete block	y	y	y	y	-	-	y	n	-	n	y	y	y	n	-	y	y	y	-	y	12
10 Save File	n	n	n	n	-	-	y	n	y	n	y	y	y	n	-	y	n	n	-	y	7
11 Rename file	y	y	n	n	-		y	y	y	n	y	y	y	y	-	y	y	n	-	y	12
Tasks attempted	11	10	11	11	-	-	11	11	10	11	11	11	11	11	-	11	11	11	-	11	
Total y	5	6	5	5	-	-	9	4	5	6	8	5	8	5	-	8	7	6	-	7	

Table 14: Performance on word processing tasks

6.2.4 Save file and return to main menu

Half of the users went straight for the SAVE & EXIT option – obviously the one they were familiar with. When prompted, most users knew that this would exit the word processor. It seems that they did not know outcome of the other two options, SAVE & RESUME or SAVE & DONE. This indicates that despite written instructions in training materials, and numerous verbal reminders during the training and practice sessions, users had only ever saved their files at the end of a instruction session, and not backed up their work as they went along.

6.2.5 Additional observations

(1) *Marking surprise:* In certain circumstances (when a block marker is inserted at the very beginning of the text, or when a block of text is still marked further down in the document), the system will mark the text before the end marker is inserted. Most users were surprised (and worried) when this happened, and none of them could explain why the system would do this.

(2) *Delete options:* Most users started to delete longer pieces of text by repeatedly pressing the DELETE key. When asked, most of them knew or guessed a quicker way of doing it, but preferred to use the simple and "safe" method.

6.3 Discussion and Conclusions

6.3.1 Effectiveness of the scenario

The observation sessions yielded on average 1.5 pages of verbal protocols per user. Since user-system interaction in this study was tightly structured, the tapes were comparatively easy to transcribe and analyse.

User	Model	Characteristics	Components
1	Computer	NOT mainframe computer storing files	disk
2	Typewriter	correct mistakes change text	
3	Document production (letter)	correct mistakes storing files	
4	Text manipulation	change text re-use text work on screen	screen, disk
7	Typewriter Filing cabinet	extensive, computerised change text store information	
8	Manipulate text	correct mistakes	
9	Computer Document production (pages)	storing information save	
10	Typewriter	on a computer storing information more flexible	
11	Document production (letters, documents)	correct mistakes storing information	
12	Typewriter	more sophisticated correct mistakes multiple copies	
13	Typewriter	more variable correct mistakes	
14	Typewriter	correct mistakes storing information layout	disk, memory
16	Typewriter	correct mistakes storing information spaces are different syntactic elements in filenames	keyboard, memory
17	Typewriter	correct mistakes change text	keyboard
18	Typewriter	really, really complex correct mistakes	
20	Document production	correct mistakes	

Table 15: Users' models of word processor

6.3.2 Evidence of users' models

All users produced some description of a word processors at the end of the session, and the statements which indicate some form of model are summarised in Table 15. The responses fall into four different categories:

- typewriter (9)
- computer (2)
- document production system (3)
- system for manipulating text (2).

More than half of the group describe the word processor as a typewriter with additional features. It seems that most users chose an existing model, of a tool they would have previously used for the same task, and added features to it. They employed a metaphor (UW) as the basis of their user model. The descriptions elicited (see Table 15), together with users' performance on the main and transfer tasks and errors, indicate that the process of transforming the metaphor into a users' model (UW $\rightarrow$ UC) did not progress past Stage 2 of the process described by Wozny (1989; see 4.2.2). The characteristics most users had added (a) were the ability to correct and change text; and (b) storage and re-use of information. The most detailed descriptions in terms of differences were provided by User 7 and 16, who are also among the top performers with 9 and 8 completed tasks, respectively. This, and the fact that so many users still stuck with the metaphor after 4 weeks of instruction and use, suggests that despite the adaptation process not having been completed, users found the metaphor a useful starting point. Only two users described the system as a *computer*. The description by User 1 as *"not a mainframe"* suggests that, even though he had *"only typed in data"* when temping, used this as a starting point. Both users managed to complete only 5 tasks, which may indicate that as a starting point, it was less helpful than the metaphor. This is not surprising since their knowledge of computers - in spite of the word *mainframe* - of 4 weeks' word processing only. Three users described the system in terms of the document production task, and two users provided a description of editing procedures only.

The problems reported with inserting text (see section 6.2.1) started because most users failed to realise that they were at the beginning of the file. The system used no

markers or symbols for beginning or end of the file, so users treat the top of the screen as the top of a physical page of paper - something you cannot change. Users could have inferred that they were at the top of the file from the line counter at the top of the text, which stated that the cursor position was "line 1", but most users paid no attention to this until they were directed at it by the investigator. Once this had been established, here was a difference between those users who had a correct representation of the effect of the CARRIAGE RETURN which, despite having the same name, is considerably more powerful than the same key on a typewriter. Those users who suggested creating space by "cutting and pasting" the text to a different location to create space seem try to transfer the operation which would have to be performed on a physical page (which are discrete entities), instead of taking advantage of the continuous space provided by a file (i.e. they attempt a U(WT) → U(CT) mapping).

From the analysis of the tapes, we can conclude that most users' models did not represent the difference between *space* as a text character and uncharted space. It is not possible to move the cursor into *uncharted space*, which looks the same on screen as a text space. From their previous practice, they had not inferred that the cursor keys only work on text, and that *uncharted space* has to be turned into text by using SPACE BAR and TAB. This is almost certainly due to the lack of visible distinction between the two types of space in the user interface.

One of the conclusions to be drawn from the transfer tasks is that in the face of uncertainty, users select familiar command names when logical inference would suggest to try an unfamiliar one. Since they know the effects of the familiar commands – in the case of the MOVE command many had just used it – logic would dictate that they should *eliminate* those commands from the list of possible options, since cannot provide the novel functionality that they are looking for. True to the predictions of Johnson-Laird's (1983) theory of mental models, however, our users did not employ logical inference mechanisms. Instead, here is clear evidence of *linguistic cueing* (e.g. COPY because the instructions said *"Copy this paragraph into a separate file"*), a reflex-like reaction caused by the recognition of a familiar word.

Similarly, it is interesting that only 4 users managed to infer the READ command in the second transfer task, immediately after they had been shown how to perform the

opposite operation using WRITE. READ and WRITE might be logical opposites for system designers, who have a model of the application program reading files off the storage medium and writing files to it (see Figure 3). For the users in this study, "reading a file" meant reading it on the screen, and they interpreted READ in that context in a "file preview" facility. This ties in with the observation (see 5.2.4) that users did not save their files during the instruction sessions. The users tend to view any changes they make on the screen as permanent – for them, the screen displays the "master" of the document, rather than the copy that it is (see Figure 4) – despite being told numerous times that changes made on screen will not appear in the document until they save it. This ties in with many instructors' observations that for many users, it takes a fairly drastic experience, such as losing several hours' work, to acquire this knowledge. Again, lack of sufficiently visible feedback between saved and unsaved files probably lies at the root of users' failure to represent this distinction in their UCs.

There is also an indication that UCs model files as *mutually exclusive entities* – you open a file to work on it, and close it if you want to work on another file. When asked to create the new file, or search for the file which they were asked to include (see section 6.2.2), most users' spontaneous reaction was to close the document they were editing. These users were surprised to learn that they could create a new file, or search for an existing file, without leaving the document they were editing. Users seem to represent document files as closed, separate units: even though they knew that they could copy and move text within a document, it did not occur to them that this can be done between files. This provides another example of a linear users' model based on paper documents (an observation made both by Marchionini (1989) and Gray (1990), see section 4.3).

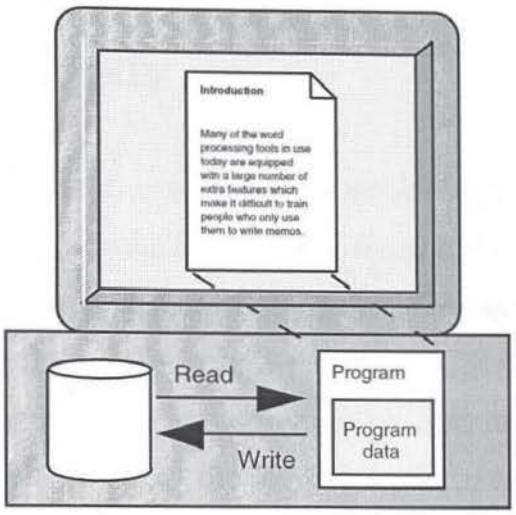


Figure 3: Designer's view of Read and Write commands

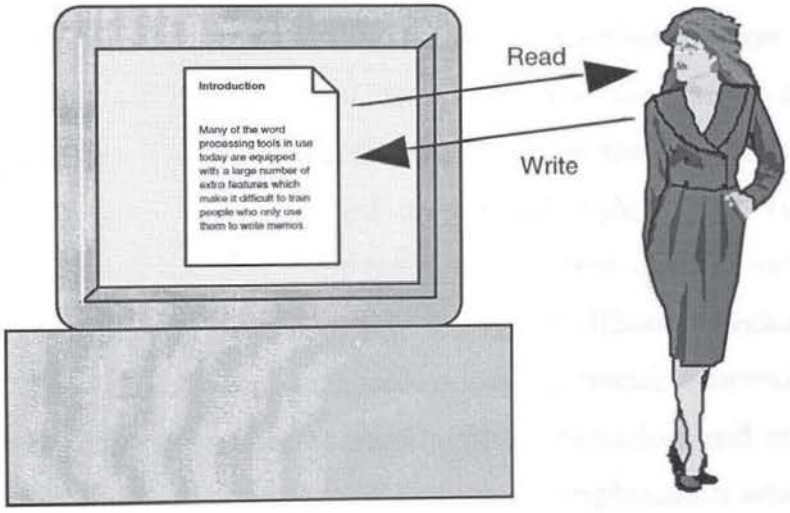


Figure 4: User's view of Read and Write commands

7 Spreadsheet Study

The purpose of this study was to observe users explaining a spreadsheet application in the context of a *teach-back* scenario. 20 users from the main group were asked to teach a new user (a co-investigator) the basic knowledge required to use the spreadsheet after 4 weeks of training and practice. Users were asked to start the session with a general explanation of a spreadsheet and its application. Since users were in charge of the interaction, the task context was not tightly structured, resulting in more variation in the number and type of tasks attempted. A comparison group of 20 users with similar experience of the spreadsheet, but a background in mathematics and computing, was investigated in the same manner.

7.1 Procedure

Users were asked by the investigator to teach another person how to use a spreadsheet application (Open Access), on which they had trained and practised in the 4 preceding weeks. This scenario is a variation of the *teach-back* approach used by Van der Veer & Wijk (1991), based on the principle of *constructive interaction* (Miyake, 1986), designed to encourage users to verbalise their knowledge about the system in (see in 5.1.2). The scenario here is slightly different because it employs a contrived co-learner (co-investigator), rather than a genuine novice; the reason is that this approach makes it easier to standardise interaction and ensure that users are prompted in an unobtrusive manner to provide explanation whenever possible. The investigator suggested to the user that the "learner" should operate the keyboard, and that users should try to teach the learner the basic knowledge required to use the spreadsheet software in one hour. The investigator would suggest that users start off with a general explanation of what a spreadsheet is, and then leave the room to watch the session on a monitor in an adjacent room. Users were told to call for the investigator if they had a problem with the system from which they were unable to recover. Users were encouraged to get the learner to set up a simple example spreadsheet; if users could not think of one, the co-investigator would suggest a simple accounts spreadsheet for a shop.

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Attempts	Total y
Enter text	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	20	20
Enter numbers	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	20	20
Enter numbers as text	y	n	-	-	n	-	-	-	-	-	-	-	n	-	y	-	-	n	-	-	6	2
Formula	n	n	n	n	y	n	n	n	y	n	y	n	y	y	y	y	n	y	n	y	20	9
Copy formula	y	y	n	y	n	y	n	-	-	y	y	-	y	y	y	-	y	y	n	y	16	12
Ins/del row/column	-	y	-	y	-	y	-	y	-	y	y	-	-	y	y	y	y	y	y	y	13	13
Change width	y	-	-	-	-	y	-	-	y	y	y	y	y	y	y	-	-	y	-	-	10	10
Filler character	-	-	-	-	-	y	y	-	y	y	y	-	y	y	y	-	-	y	-	-	9	9
Change attributes	y	-	y	y	-	y	-	-	y	y	-	-	y	y	y	y	y	n	n	y	14	12
Toggle display	-	-	-	-	-	y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1
Blank entries	-	-	n	-	-	-	-	y	-	y	-	-	-	-	-	-	-	-	-	-	3	2
Save	y	y	y	y	y	y	n	y	y	y	y	n	y	y	y	y	y	n	y	y	20	17
Tasks attempted	8	7	7	7	6	10	6	6	7	10	8	5	9	9	10	6	7	10	7	7		
Total y	7	5	4	6	4	9	3	5	7	9	8	3	8	9	10	6	6	7	4	7		

Table 16: Performance on spreadsheet tasks

The co-investigator would also propose to do certain tasks (such as working out profits) to ensure that users tried to address key tasks such as entering and copying formulae.

7.2 Results

7.2.1 Results of the main group

The transcripts are provided on pages 23-67 of the Transcripts volume. 20 users participated in this observation, and Table 16 provides an overview of the tasks attempted and completed. 12 different tasks were tackled altogether, with each user attempting between 5 and 10 tasks. The tasks which most users failed to complete were:

- entering formulae (11 fails of 20 attempted); and
- handling of text and numerical entries (4 fails of 6 attempted).

Formulae

All users attempted this task (urged by the co-investigator who was keen to do at least one calculation), but eleven failed. There was a variety of reasons for this:

- failure to construct a formula (subtraction or multiplication) (4);
- looking for a "formula" command in the menu options (2);
- problems with syntax (but identified correct formula) (2);
- failure to recall the name of the formula (1);
- failure to locate key to toggle formula to NUMERICAL (1)
- entering result of calculation "by hand" (1).

Entering numbers as text

All users managed to enter simple text and numbers into the spreadsheet. 6 users, however, chose alpha-numeric labels for column headings in which the first character was a number. All six tried to enter the label as a number, which resulted in an error. Subject 15 (see Transcripts-54) was the only one to diagnose the cause of the problem immediately, and to toggle the entry to text before entering it. Subject 1 (Transcripts-23) diagnosed the cause of the error on the second attempt and applied

the toggle function, but then proceeded to enter his data as text, too. He eventually realised that this was a problem when he tried to apply a formula, and re-entered the numbers correctly. Subject 2 (Transcripts-25) tried to enter the date ("1-12-88"). The system interpreted this as a numerical entry, and displayed the result of the calculation ("-99"). He tried to apply the toggle function, but since the toggle has to be applied *before* the entry is entered into the cell, this did not work. He decided to enter a text label (January) instead. Subjects 5 and 13 (Transcripts-32 and 49) could not work out the cause of the error, but once the co-investigator managed to direct their attention towards the "NUM" indicator on the status line, they applied the toggle key. Subject 18 diagnosed the problem correctly, but it did not occur to her that she might apply the toggle key, and she circumvented the problem by changing the label ("Hundred m").

Additional observations

(1) *SAVE uncertainty:* Even though 17 users chose the correct procedure of exiting the spreadsheet to save, most of them were convinced they had not saved it. After selecting QUIT, they expected a prompt "*Quit with or without saving*"- which the system would do when a existing file had been edited. Since the model had not been saved previously, the system automatically saved the model when users left it, but did not give any message or feedback to this effect.

(2) *Moving cursor during entry:* Several users forgot to position the cursor in the target cell before entering, for instance, a formula. The spreadsheet allowed cursor movement during most operations, but entries or results would be entered in the cell on which the cursor was positioned when the first character was typed. When this happened, users stated, somewhat vaguely, "*it doesn't like that,*" but proceed to position the cursor correctly and repeat the operation (except for user 13, who entered the – incorrect – result manually in the target cell instead of repeating the formula). None of them gave any indication that they realised that the cursor position at the first keystroke was critical, but since the co-investigator did not probe them on this matter, we cannot be sure.

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Attempts	Total y
Enter text	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	20	20
Enter numbers	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	20	20
Enter numbers as text	-	-	-	y	-	y	y	y	-	-	n	-	-	-	-	-	-	-	y	y	7	6
Formula	y	n	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	20	19
Copy formula	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	20	20
Ins/del row/column	-	-	n	-	y	-	y	y	y	y	y	-	-	y	-	y	y	-	-	y	11	10
Change width	-	-	-	-	-	-	-	-	-	y	-	y	-	-	-	-	-	-	-	-	2	2
Filler character	-	-	-	-	-	-	-	-	-	y	-	-	-	-	-	-	-	-	-	-	1	1
Change attributes	y	n	y	-	-	y	-	y	n	y	n	y	y	n	-	y	-	-	-	-	12	8
Toggle display	-	y	-	-	-	y	y	y	-	-	-	-	y	y	y	-	y	-	-	-	8	8
Blank entries	-	-	-	-	-	y	-	y	y	-	-	-	-	-	-	-	-	-	-	-	3	3
Recalculate	y	-	y	y	y	y	-	-	-	y	y	y	y	y	y	y	y	-	-	-	13	13
Save	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	20	20
Tasks attempted	7	7	8	7	7	10	8	10	8	10	9	8	8	9	7	8	8	5	6	7		
Total y	7	5	7	7	7	10	8	10	7	10	7	8	8	8	7	8	8	5	6	7		

Table 17: Performance of comparison group on spreadsheet tasks

7.2.2 Results of the comparison group

Twenty users participated in the comparison group, attempting 13 different tasks overall. The transcripts are provided in Transcripts 68-128. Table 16 provides an overview of the tasks attempted and completed. All but one user managed to enter formulae, and six out of seven users who attempted to enter numbers as text labels changed the label before entering

Additional observations

(1) Most users in this group started the session with a lengthy explanation, and often demonstration, on the difference between entering numbers and text, and the application of the toggle key.

(2) Most users in the comparison group entered formulae by specifying the calculation, rather than using the built-in formula commands such as SUM and MEAN (for instance Comparison-Subject 3, Transcripts-73; Comparison-Subject 4, Transcripts-75).

(3) Users in this group demonstrated the toggling between displaying numbers and formulae without being prompted, and changed numbers to demonstrate the RECALC function.

7.3 Discussion and Conclusions

7.3.1 The effectiveness of the scenario

The *teach-back* scenario employed in this study is a further development of the *constructive interaction* (Miyake, 1986) technique employed in the fifth study. The success or failure of this scenario obviously depends to a large extent on the skills and behaviour of the co-investigator. Users have to be fully convinced that the "learner" does not know anything about the system – otherwise they are not motivated to communicate their knowledge to the co-investigator. Good typing skills or knowledge about the layout of the keyboard can have this effect. To avoid this happening, the co-investigators were instructed to type with one hand only to slow them down, and to ask *"Which one is that?"* or *"Is it this one?"* when users mentioned a key.

In terms of verbal protocols, this scenario yielded on average 2 pages for the users in the main group, and 3 pages for users in the comparison group. Users in the main group tended to give entirely procedural instructions (*"Select this option, now press that key"*). With a "learner" as a co-investigator, prompting for further explanation of procedures (*"Why do I have to do that?" "How does this differ from command X?"*) is a fairly natural thing to do. The scenario therefore offers a good chance to elicit information about how such procedures are represented, without changing the demand structure of the interaction situation. Users in the comparison group gave quite a lot of explanation without prompting. The co-investigator very cautiously tried to prompt the users in the main group by demanding explanations whenever appropriate. Training co-investigators not to betray their actual knowledge in such scenarios is crucial, and despite training, keeping up an appearance of being new to the system turned out to be very hard work.

Constructive interaction scenarios can only be analysed from recordings. Transcribing and analysing the observations from these studies required a lot more time and effort, more so than in the structured scenarios in which users attempted a pre-defined set of tasks in a certain order. Co-investigators can exert a certain amount of influence to ensure that the tasks in which the investigator is interested in are attempted. If, however, the co-investigator strongly insists on attempting certain tasks, the user may be deprived of their leadership role and become more passive, thus revealing less about their users' models. Similarly, if the co-investigator had urged users to provide the general explanation of the spreadsheet suggested by investigator, users' may have relinquished initiative. This was also the reason why the co-investigator did not insist that users provide the general explanation suggested at the start of the session (see below).

7.2.3 Evidence of users' models

The investigator's suggestion that users provide a general explanation of the spreadsheet to the learner was not followed by all users; many started by setting up an example speadsheet without giving any introductory explanation to the co-learner. The statements provided by the users in the main group are presented in Table 18, and in Table 19 for the comparison group. The explanation most

frequently given by users in both groups refers to the surface structure (appearance) of the spreadsheet:

- rows and columns, or coordinates and their labelling (6 in main, 8 in comparison group)
- matrix (2 in comparison group)
- table (2 in comparison group).

User	Explanation
2	big square rows (1,2,3 ...) and columns (A,B,C ...)
3	microfilms in library: big sheets with lists on bigger than the screen
4	programming information in the computer like writing an essay: put in words and move them about
7	rows and columns
9	calculation basically like word processing: presentation
10	co-ordinates (spaces which have)
13	storing information calculation rows and columns (infinite number of)
18	rows and columns
20	computer changing information graphs

Table 18: Users' explanations of Spreadsheet (Main Group)

A different aspect of the spreadsheet, also related to the display, mentioned by several users is the fact that the what is seen on screen is only window on a much bigger sheet:

- big square (1 in main group)
- bigger than the screen (1 in main group)
- like a microfilm in a library (1 in main group)

- like a big sheet of paper (3 in comparison group)

User	Explanation
1	rows and columns (array of) fill with text, numbers, formula computerised table paper (big sheet of)
2	putting numbers in text, numbers, formula
3	rows and columns (numbered) text, numbers, formula
4	referenced blocks
5	rows and columns (as many as computer has memory)
6	matrix (also sketched on paper) battleships: move around using location co-ordinates
8	rows and columns (list of) text, numbers, formula
9	table to manipulate data
10	rows and columns
11	matrix paper (big sheet of)
12	rows and columns calculation (sums)
13	rows and columns calculation (averages)
15	table of data
16	equations variables presentation calculation
17	rows and columns
18	calculation (lots of; sums, percentages)
20	paper (huge sheet of) co-ordinates

Table 19: Users' explanation of Spreadsheet (Comparison Group)

The main function of the spreadsheet, calculation, is mentioned by only two 2 in the main, and 4 in the comparison group. Two users in the main group make a reference to the only other application they have experience with (word processing).

Users in the comparison group provided considerably more explanation to the "learner" than the users in the main group. They tried to explain the behaviour of the system and why commands had to be used in a certain way. Users in the main group gave mainly procedural instructions required to get the example spreadsheet set up, and very little additional explanation, unless prompted by the co-investigator.

From the descriptions given, it seems that the *visual appearance* of the spreadsheet has made a great impact on users' models in both groups. It is mentioned far more frequently than the principal *function*, calculation. Thus, the overriding impression users have is what the system looks like, rather than what you can do with it. This provides some indication of the impact of that the system image can have on users' models (C+MC → UC).

Yet, the results and analysis of the transcripts indicate that users in the two groups had very different users' models, due to a difference in knowledge and experience (UW).

Users' in the main group do not show a model of the system beyond the visual appearance, and in particular not of its primary function. The instructions given to the co-investigator are almost entirely procedural (*"First type a number here, then you press this button, then on to ..."*). Many were trying to avoid using calculation functions altogether, and most users preferred to use and talk about the formatting functions, such as *changing the width of columns* and *inserting filler characters*. (If the co-investigator had not "nagged" some of the users in the main group to perform at least one calculation, some users would have spent the entire instruction time on fiddling with the layout of the spreadsheet.) The formatting functions are similar to the word processing functions they had previously used, even though they are executed in a very different way. Two users explicitly described the system as similar to the word processor. This suggests that users in the main group were drawing on the users' model of the only other system they knew at this state (UC1 → UC2). It can be argued that the reason for defaulting to this model is that, unlike

in the word processing study (see Chapter 6), many users in the main group had problems did not have a suitable model of a real-world task (U(WT)) onto which available functions could be matched. Many users in the main struggled to identify a task for the "learner" to work on, and replicated an example for their course exercises or waited for the "learner" to suggest one. Interesting exceptions are users 1 (see Transcripts-23), 2 (see Trans-25) and 15 (see Trans-54), who formulated conceptually similar tasks from their subject of study (physical education) for their example spreadsheet. They also embarked on calculations out of their own initiative. This indicates that they had identified a real-world task from their general knowledge and experience, and could map it onto the primary functionality provided (UW → U(WT) → U(CT)). Most other users recalled some procedures (U(CT)) but had problems specifying sub-goals in natural language (which indicates incomplete U(WT), or make the mapping between them (U(WT) → U(CT)). Thus, their model was based on the visual structure of the system image, and functionality from the word processing system (as illustrated in Figure 5).

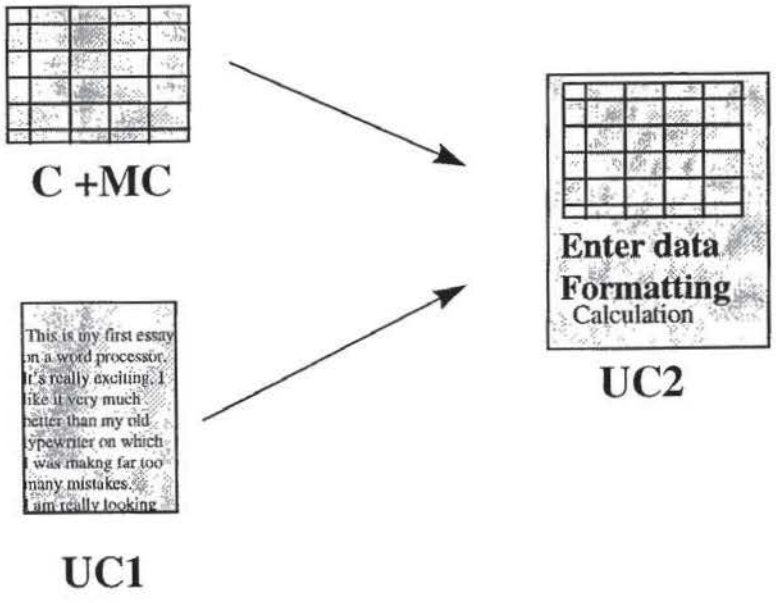


Figure 5: User's model (UC) in main group

Most users in the comparison group, who had a background in maths and programming, had no problem finding an example, and the real-world tasks they formulated very more sophisticated (e.g. calculating the trajectory of a rocket)- another indication of the importance of general background knowledge for

formulating the task (UW→ U(WT)). 13 users demonstrated the RECALC function (whereas none in the main group did). On the other hand, few users in the comparison group bothered with the formatting commands, inserting dashes instead of using *filler characters*. Even though users' overall performance in the comparison group is better, and they were considerably more confident then users in the main group, one could argue that their ability to match the high-level function to their existing knowledge and experience resulted in them using the spreadsheet in a sub-optimal fashion even where *calculation* is concerned. Many users in the comparison group specified their own formulae instead of using built-in ones (e.g. b4+b5+b6/3 instead of using MEAN), thereby creating extra work for themselves. It is likely that, since these users are accustomed to specifying these calculations in their programming work, they transfer this method to working with a spreadsheet (UW → UC). It may be that having identified a solution, they do not consider the functionality of the system. Either their model of the system functions (U(CT)), or they do know about system functions, but prefer to trade off the physical effort involved in typing out a calculation against the cognitive effort of remembering the name and syntax of the formula. The latter would be an example of what Norman (1983) characterises as *parsimonious* behaviour (see 4.1.3). Unfortunately, the co-investigators did not prompt users in the comparison group on this matter. An illustration of how many users' model in this group may have been derived is presented in Figure 6.

Another indication of a marked difference in users' models (UCs) in the two groups, again due to differences in users' background (UW), is the way in which they handle TEXT and NUMBERS. Most users in the comparison group started with a lengthy explanation of the difference between text and numerical entries, and demonstrated the use of the TOGGLE function[35]. Users in the main group only started to explain this when they encountered a problem when entering a formula, or entering text containing numerical characters. Together with difference in performance on

---

[35] The system would label any entry typed into a cell as TEXT or NUMERICAL, depending on whether the first character is a letter or a number. The TOGGLE function has to be used to change this default classification if a text label starts with a number (e.g. "100 meters"), or a numerical entry starts with a letter (e.g. "MEAN A4:F4").

formulae (all but one user in the comparison group applied formulae successfully, whereas only 9 users in the main group did), and on *entering numbers as text* (6 out of 7 successful in the comparison group, 2 out of 6 in the main group), this suggests an important difference in users' models. To users in the comparison group, text and numbers are fundamental characteristics of entries, and something which has to be considered and declared. All further operations are represented as depending on that the type of data entered. This users' model of how the system handles text and numbers greatly improves their competence when interacting with the system: these users displayed an ability to explain procedures to the learner, and to diagnose correctly when they had forgotten to toggle an entry. It can be assumed that the users' model which helped them so much is a result of their programming experience, where data types have to be declared; this is another example of a relevant model from their general knowledge which they applied to the spreadsheet system. Users in the main group, on the other hand, have a "face value" model of text and numbers: they do not consider the need to distinguish between the text and numerical entries until they encounter a problem. Then, the toggle function is applied as a procedural afterthought. Some users in this group managed to diagnose the problem correctly, and knew that the use of the toggle key was part of the solution, but still had problems applying the toggle function correctly to recover from the particular problem which they had created. Text and numbers are essential parameters in this system, and the observation confirms that users' models (UCs) which do not represent these essential building blocks correctly do not support user-system interaction adequately (Payne et al., 1990, see 4.2.5).

In summary, whilst both groups' users' models seem to include the system image, the differences in performance and the verbal data indicate significant differences in users' models (UCs). Users in the comparison group seem to draw on their general knowledge and experience (UW) to formulate the task-component (U(WT)) of the users' model: they see the spreadsheet application as a tool for number-crunching and formulate real-world tasks (U(WT) accordingly. They make an explicit distinction between text and numerical data. At the same time, their model of system functions relevant to that task (U(CT)) is either incomplete, or users prefer to draw on their genera knowledge and experience (UW) in preference to U(CT).

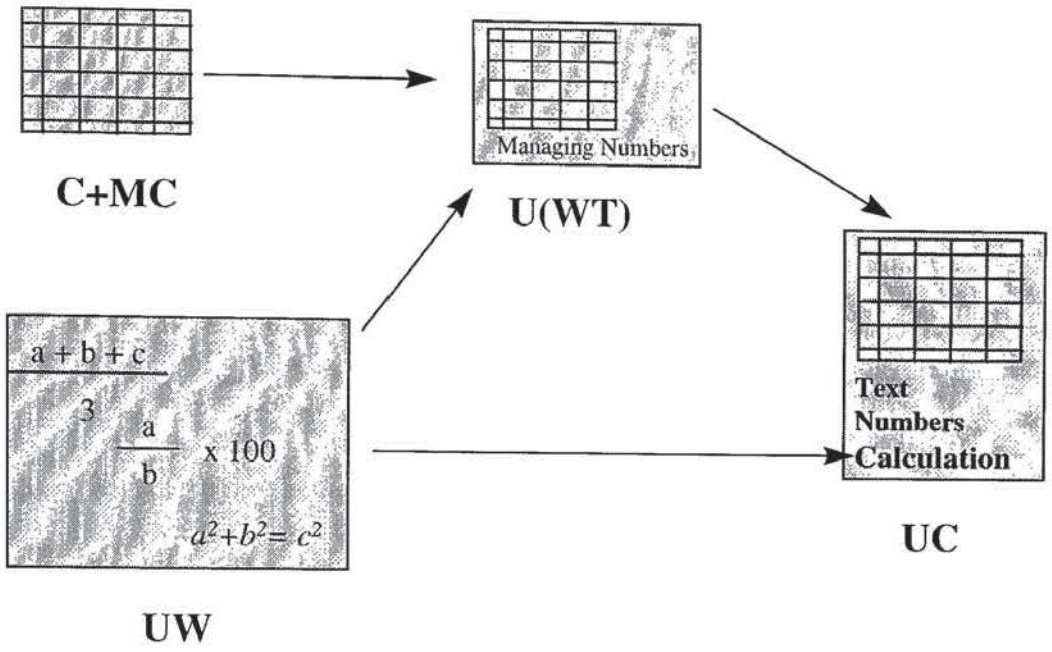


Figure 6: User's model in comparison group

The models of users in the main group are at least partly influenced by models of the word processing system (UC1 → UC2). Many seem to have a model of the real-world task (U(WT)) which is dominated by formatting, with a small element of calculation: to them, the spreadsheet is an application for producing pretty-looking tables. The mapping between U(WT) → U(CT) functions seems to function well enough for the formatting tasks, but due to absence of a similar model for calculations, their representation for executing these functions is basically in procedural format.

8 Database study

The purpose of this study was to check users' knowledge of the terms used to describe a database application. 18 users participated in a quiz-style dialogue in which the investigator pointed to parts of the display and asked for the term by which it was described in the instruction material, before performing 2 retrieval tasks and a transfer task. Users were also asked to compare the database application to the spreadsheet application investigated in the previous study.

Terms	**Description**
1 File	Could you please describe what a database file is?
2 Field	Do you know what one of these columns is called? Can you describe what a field is?
3 Record	Do you know what one of these lines in the file is called? Can you describe what a record is?
4 Entry	Do you know what one of those cells is called? How would you describe what an entry is?
5 Diff. field/record	How would you describe the difference between fields and records?
6-10: Data Types	What different types of data can you have?
11-14: Query Language	You have used a query language to retrieve information from files. Can you remember what the query language consists of?
15-17: Addresses?	What do the commands FROM, SELECT, WHERE refer to?
Tasks	**Description**
18 Task 1	You want a list of all customers names and postal addresses for a mailshot.
19 Task 2	You want a list of all people whose payments are overdue, the name of a contact person and their phone number, so that you can start chasing outstanding payments
20 Task 3 (TT)	You want to find out which customers who have purchased how much from you, and list them by the size of their purchases.
21 Difference DB/SS	Could you describe the difference between the database and the spreadsheet

Table 20: Database tasks

8.1 Procedure

This study was not directly based any of the observation scenarios suggested by Young (1983), but tried to illuminate the relationship between semantic knowledge that users hold on the one hand, and their performance on the other. Semantic knowledge was elicited by asking users questions about the system, and subsequently observing them using the system (the Open Access database). The study therefore consisted of three parts:

(1) The investigator (the author) asked the users 17 questions about the database (see Table 20), checking whether the users knew the basic terminology the commands of the query language. All the terms had been introduced in the tutorials, and were explained on the instruction material given to users for this application users. Users were allowed to view an example file while answering those questions.

(2) Users were then presented with three tasks, for which they had to construct and execute a query on a given database file. The third task required sorting the records retrieved in descending order, which was a transfer task.

(3) Users were asked to describe the difference between a database and a spreadsheet.

8.2 Results

18 users participated in this study. The verbal protocols of the observations on this scenario are provided in Transcripts 114-127, and an overview of the results is given in Table 21. The findings for each group of tasks are given below.

8.2.1 Terms

All users could give some description of a *file* or could identify files in the system. 60% of users did recall the term *field*, or knew that this was referring to the columns in a file. Half of the users recalled the term *record*, or knew that this referred to the rows in a file.

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Total y
Terms																					
1 File	y	-	y	y	y	y	y	y	y	y	y	y	y	y	y	y	-	y	y	y	18
2 Field	n	-	n	n	y	n	y	y	y	y	y	y	y	n	n	y	-	y	y	y	12
3 Record	n	-	n	n	n	y	y	y	n	y	y	y	y	n	n	y	-	y	n	n	9
4 Entry	n	-	n	n	y	n	n	y	n	n	y	n	n	n	n	y	-	n	n	y	5
5 Diff. field/record	y	-	y	y	y	y	y	y	y	y	y	y	y	n	n	y	-	n	y	y	15
Data Types																					
6 Text	y	-	y	n	y	y	n	y	y	y	y	y	y	y	y	y	-	y	n	y	15
7 Numerical	y	-	y	y	y	n	y	n	y	y	y	y	y	y	n	y	-	y	n	y	14
8 Decimal	n	-	n	n	y	y	n	y	n	n	y	n	y	y	y	y	-	y	n	n	9
9 Date	y	-	y	y	y	y	y	n	y	y	y	y	y	n	y	n	-	y	y	y	15
10 Binary	n	-	n	n	y	y	y	n	n	y	y	n	y	n	n	y	-	y	n	n	8
Query Language																					
11 From	y	-	y	n	n	n	y	y	y	y	y	y	y	y	y	y	-	y	n	y	14
12 Select	n	-	y	y	y	n	y	y	y	y	y	y	y	y	y	y	-	y	n	y	15
13 Where	n	-	y	n	n	n	y	y	n	y	y	y	y	y	y	y	-	y	n	y	12
14 Order	n	-	n	n	n	n	y	y	y	n	y	y	y	n	y	y	-	y	n	y	10
Addresses ?																					
15 From – file	y	-	y	n	y	y	y	n	y	y	y	n	y	y	y	y	-	y	y	y	15
16 Address – fields	y	-	y	y	y	y	y	y	y	y	y	n	y	y	y	y	-	y	y	y	17
17 Where – cond.	y	-	y	-	-	n	n	n	y	y	y	n	y	n	y	y	-	y	-	n	9
Tasks																					
18 Task 1	y	-	y	y	y	y	y	y	y	y	y	y	y	y	n	y	-	y	y	y	17
19 Task 2	y	-	y	n	y	n	y	n	n	y	y	n	n	n	n	y	-	n	n	n	7
20 Task 3 (TT)	n	-	n	n	n	n	n	n	n	n	n	n	n	n	n	n	-	y	n	n	1
Tasks attempted	20	-	20	19	19	20	20	20	20	20	20	20	20	20	20	20	-	20	19	20	
Total y	11	-	13	7	14	10	15	13	13	16	19	12	17	10	10	18	-	17	7	14	

Table 21: Performance on database tasks

Less than 30% recalled the term *entry*, or what it referred to. When asked to explain the difference between fields and records, most users knew that fields only contain one type of data, whereas records can contain a range of data.

8.2.2 Data Types

Most users could identify *text/alphanumeric* and *dates* as data types. 10 users identified either *numerical* or *decimal*, but not both, which suggests that they used either term for numbers in a generic sense. 8 users identified *binary* choice as a data type.

8.2.3 Query Language/Addresses

Most users could recall the commands FROM and SELECT, and knew that they addressed files and fields. About half could recall the command WHERE and ORDER, and knew that WHERE was used to specify conditions.

8.2.4 Tasks

All but one user was able to complete the first task without assistance, but only seven managed to complete the second task. The problem that most encountered was to specify the condition WHERE OVERDUE = TRUE: most users just selected the field OVERDUE, and only specified the condition after prompting from the investigator (e.g Subject 8, Transcripts-119; Subject 9, Transcript-120).

The third task was a transfer task, which required users to list the retrieved records in descending order (the system default for sorting numerical or decimal fields was ascending). Only one user managed to complete the task without help or prompting.

8.2.5 Additional observations

At the end of the session, users were asked to compare database and spreadsheet applications. Only one user (Subject 14, Transcripts-124) provided an answer which summed up the functionality of the two systems accurately. About half the users mentioned that the spreadsheet could be used for calculations, and the database for selective retrieval.

8.3 Discussion and Conclusions

8.3.1 The effectiveness of the scenario

It was originally planned not to make the example file available until users were asked to perform the retrieval tasks. The first two subjects in this study could not answer any of the first 10 questions, and asked if they could look at a file. In the interest of eliciting more than "no" the investigator allowed them do so, and offered this to the other users at the beginning of the session. All users loaded the file during the first 5 questions, confirming Payne's (1991b) observation about the users' dependency on the display.

Average length of the verbal protocols elicited in this study was 3/4 of a page. Half an hour was scheduled for each session, but most users completed their session in about 20 minutes. The sessions were easy to transcribe and score (this could have been done during the session by an additional observer). By eliciting semantic knowledge as well as observing users working to actual tasks, the scenario provides an opportunity to (a) illuminate the difference between user's knowledge and competence; and (b) to gain some appreciation about the importance of certain features of the system image for guiding the user through the interaction. At the end of the session, users were asked to compare the database application with the spreadsheet used in the previous study.

8.3.2 Evidence of users' models

The most interesting finding from users' comparison of the database and spreadsheet applications pertains to the spreadsheet rather than the database application investigated in this study. Whereas only two users mentioned calculation as an important feature when asked to explain the spreadsheet in the previous study (see Chapter 7), 10 of the 15 user who answered this question mention *calculation* to contrast it to the database. Here, most users describe the main functionality of both applications fairly accurately (*storage* and *retrieval* of information for the database), rather than talking about surface characteristics. The visual structure of the displays for both applications is very similar (rows and columns), and thus offers little ground for distinguishing between the two. It is very

likely that users drew on their users' models of the spreadsheet application when they started using the database in the vein of an analogy (UC1 → UC2).

The discussion of analogy and metaphor in 4.2.2 (see also Table 10) revealed the importance of identifying and representing differences between source and target models accurately. The similarity of the applications at the surface should create the potential for confusing the two (see also Norman's (1983) observations in 4.1.3); at the same time, it may create a greater need to identify differences between the source and target model, and represent them in both models. This means that not only the user's model of the new application, but also the model that is source of the mapping, are refined during the process: (UC1 → UC2 ↔UC1 → UC1*).

Whilst their users' models have become more refined as far as functionality is concerned, half of the users could not identify two of the five data types in the example file, even though they had been introduced to them, and they had entered and retrieved all the different types of data in their previous exercises. As in the spreadsheet study, it seems that many users still have a general knowledge and experience (UW) representation of numbers and text, as discussed before in 7.2.3. Anything composed of letters, including binary data (YES/NO or TRUE/FALSE) is subsumed under text, and all numbers are just that (not differentiated into integers and decimals). The data type DATE, which is composed of a sequence of numbers, was recognised by most users as a separate data type, but this is consistent with general knowledge. Despite instruction and practice undergone, it seems that users did not refine their model from general knowledge and experience into a user's model (UW → UC).

User	Database	Spreadsheet
1	Constantly updated	For stable data (updated once-a-year)
5	Can't do calculation (averages)	Information is "trapped" Keep adding information rather than extracting
6	Storage/retrieval of information	Selects specific information More control: view whole whilst changing entries on screen
7	Stable Query language: moving between files Extracting information	Easier to create files More manipulation possible Calculation (can do maths)
8	Similar: don't see why you need two different systems	
9	Easier Storing information	Presentation Calculation
10	Retrieve information	Calculation (averages and means) View: can see what you're doing
11	Selecting information Asking specific questions	Calculation (formulae) See everything at the same time
12	Call up information Can work on bits (data entry)	Calculation (averages) Can't select individual entries
13	Location information	Call up individual records Keep track of data Calculation (maths stuff)
14	Store information	Calculation
15	Select information Separate data entry function	Type on spreadsheet itself Calculation
16	Storing information Wider: more things happening	Calculation
18	Review information Order information	Presentation Calculation
20	Storing information	Storing information Calculation

Table 22: Users' models of spreadsheets vs. database

The fact that 11 users failed to complete the second task may demonstrate linguistic cueing in action: the instructions asked users to retrieve all customers who were overdue. Half of the users realised that a condition statement was required. The other half thought that selecting the field OVERDUE was sufficient, and suggested that the records which had the value TRUE should be noted down from the screen display. They added a qualifier – which means that the system will only retrieve the records of customers who were overdue – only after prompting from the investigator. The use of qualifiers is based on Boolean Algebra, and requires that users remember a particular syntax. Like users in other studies (Borgman, 1986; Marchionini, 1989), these users had problems recalling and applying these features despite instruction and practise.

Comparing users' semantic knowledge and their competence revealed that their semantic knowledge of the most basic terms was anything but complete. They definitely did not have anything approaching a surrogate model to "run", and found it very difficult or impossible to talk about the system at all without being able to look at the display. Considering the state of their semantic knowledge, users' performance on the tasks was surprisingly good: Guided by certain cues from the system, they could recall and execute familiar procedures successfully. They were less successful on the second task, which required recalling special syntax of the query language - for these features, no cues were available from the system image. The fact that only one user managed to complete the transfer task confirms that the knowledge about the system was mainly represented as procedures rather than a model, since they failed to infer the new procedure from the knowledge they had.

9 Prolog Study

The purpose of this study was to elicit users' models of a logic programming language (Prolog) by asking them to predict the behaviour of a program and asking them to run queries and make small modifications to the program. 18 users participated in the session after 5 weeks of training and practice with the programming language.

Predictions	Could you please predict, after looking at the program listing, which answer the system will return when you type in the following queries:
1 p(o,o)	person(annie, female).
2 p(V,o)	person(X,female).
3 p(o,V)	parent(jenny, X).
4 p(o,o)	parent(sophie, rupert).
5 p(o,o)	parent(carol, rosie)
6 p(o,V)	parent (rosie, Y).
7 Conjunction (TT)	What query would you have to enter to get a list of all daughters in the database?
8 Test rule	Please examine the father rule. Which query, using that rule, would give you a list of all fathers and their children?
9 Place holder	What query would you have get the system to list you only the names of the father, without returning the names of the children?
10 Repetition	Why does the system return some names more than once in response to your query?
11 Analogous rule	Based on the father-rule, write a rule that defines mother.
12 New rule (TT)	Write a rule that defines either brother or sister.

Table 23: Prolog tasks

9.1 Procedure

18 users participated in the fourth study, in which they were asked to predict the behaviour of a PROLOG program (Edinburgh Prolog - see Appendix 4 for a listing). At the time of the study, the users had had 5 weeks' instruction and practice with Prolog, and had written a small application (a specialist version of Talking Pages), working in teams of two. In the half-hour session, the investigator (the author) presented them with a short Prolog program on the computer, consisting of a knowledge base and one rule. The users could also refer to a printout of the program. The study consisted of three sets of tasks (see Table 23 for an overview):

(1) To start with, the investigator asked the user to type in a number of PROLOG queries, and predict the answer the system would return before entering the question. If the answer returned by the system did not match the user's prediction, the investigator asked the user to work out why the program returned a different answer.

(2) The investigator then presented 4 questions in natural language, and asked the user to construct the PROLOG queries required obtain answers to these questions. The user then proceeded to type the queries in. If the query did not return the predicted answer, the investigator asked the user to work out the reason for the system returning the answer it did. Question 7 could only be answered by entering a *conjunction*. The users had constructed such queries during their classroom work, but since they had to work out a conjunction was required to answer this question, this question was regarded as a transfer task.

(3) Finally, the investigator asked the user to write two new rules and add them to the program. One of those rules was analogous to the one which they had tested previously, the second one was new and therefore a transfer task. (The users had written rules of much greater complexity during their PROLOG assignment). The user was then asked to explain what the rule would do, and run a few queries to check if the rule was working as intended. If the rule did not work, the investigator encouraged the user to try and explain why it had not worked.

9.2 Results

18 users participated in this study. The transcripts are provided in Transcripts 129-167, and an overview of users' performance on the tasks is shown in Table 24. Most users could predict the results of the queries correctly (though some needed warming-up practice on the first prediction) and could construct an analogous rule. Half of the users could construct a query to test a given rule, and knew how to use a placeholder variable. Users did, however, have considerable problems on the transfer tasks.

9.2.1 Conjunction

4 users mentioned the term *conjunction* at the beginning of this task, but one of them then proceeded to write a rule. 4 users suggested writing a rule rather than entering a conjunction. Some users suggested entering the first part of the query, writing down the results of that query, and then match the results from the second query (e.g. Subject 2, Transcripts-132). Most users managed to work out one part of the rule (person(X, female).), but then had problems to formulate the second condition. This was not just due to problems expressing the second condition (parent(_, X).) in PROLOG, but several users had difficulty to identify this condition in natural language (e.g. Subject 1, Transcripts-129, Subject 12, Transcripts 151).

9.2.2 New rule

None of the users managed to write the new rule (defining *brother* or *sister*) without help from the investigator. Most subjects managed to work out the header of the rule, and the condition that the brother had to be male. The other two conditions (that they had to have a parent, and that there had to be another person who had the same parent), posed insurmountable problems for the users. Some users needed repeated prompting from the investigator before expressing those conditions in natural language (e.g. Subject 1, Transcripts 129; Subject 7, Transcripts-143). Most users managed to express the conditions in natural language, but needed help with the translating the "same parent" part into a condition (e.g. Subject 11, Transcripts-148, Subject).

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Total y
Predictions																					
1 p(o,o)	n	y	y	y	y	y	y	-	n	n	y	y	n	n	n	y	y	-	n	y	11
2 p(V,o)	y	y	y	y	y	y	y	-	y	y	y	y	y	y	y	y	y	-	n	n	16
3 p(o,V)	n	y	y	n	y	y	y	-	y	y	y	y	y	y	y	y	y	-	y	y	16
4 p(o,o)	y	y	y	y	y	y	y	-	y	y	y	y	y	y	y	y	y	-	y	y	18
5 p(o,o)	y	y	y	y	y	y	y	-	y	y	y	y	y	y	y	y	y	-	y	y	18
6 p(o,V)	y	y	-	y	y	y	y	-	y	y	y	y	y	y	y	y	y	-	-	y	16
7 Conjunction (TT)	n	n	-	n	y	n	n	-	n	n	y	n	n	n	n	n	y	-	-	n	3
8 Test rule	n	y	y	n	y	n	y	-	y	n	n	n	y	n	y	y	y	-	n	n	9
9 Place holder	n	n	-	y	n	n	y	-	y	y	y	y	y	y	n	n	n	-	-	n	8
10 Repetition	y	y	y	y	y	y	y	-	y	y	n	n	y	y	y	n	y	-	-	n	13
11 Analagous rule	y	y	y	y	y	n	y	-	n	y	y	y	y	y	y	y	y	-	n	y	15
12 New rule (TT)	n	n	n	n	n	-	n	-	n	n	n	n	n	n	n	n	n	-	n	n	0
Tasks attempted	12	12	9	12	12	11	12	-	12	12	12	12	12	12	12	12	12	-	8	12	
Total y	6	9	8	8	10	7	10	-	8	8	9	8	9	8	8	8	10	-	3	6	

Table 24: Performance on Prolog tasks

9.2.3 Additional observations

(1) *Objects and variables:* Some users had could not explain the nature of purpose of objects and variables (e.g. Subject 11, Transcripts-150). This was quite astonishing given the amount of practice they had had in writing and querying their own program.

(2) *Variable names:* A related observation is that quite a few users believed that they had to use the same variable name when querying a rule, as the one employed in the rule itself (e.g. Subject 2, Transcripts-133). There was also a belief strong that variable names had semantic meaning (e.g. Subject 6, Transcripts-141).

9.3 Discussion and Conclusions

9.3.1 Effectiveness of the scenario

The verbal protocols elicited had an average length of 2 pages per user – however, a lot of the talking was done by the investigator rather than the users. This was mainly due to the fact that users had a lot of problems with the transfer tasks, and the investigator had to spend a lot of time explaining why rules suggested by the users would not work. In general, asking users to predict system behaviour prompts thinking aloud in a fairly natural way when errors occur. Altogether, though, this was probably the study which generated least insights of users' models.

9.3.2 Evidence of users' models

The most striking observation in this study was that, whilst users were able to use the program fairly competently, they did not seem capable of applying that knowledge to a program of a different real-world task. After five weeks of programming instruction and practice with the programming language, which included writing a moderately large program for a Talking Pages-style enquiry service, users might have been expected to be able to write a comparatively simple rule. Alas, this expectation turns out to be an incarnation of the formal equivalence fallacy: ability to construct a rule does not transfer from a program to deal with one

task to another. The task in this study was chosen because it could be assumed that all users would have the necessary general knowledge and experience (UW) to formulate in natural language what makes someone a brother of somebody else; however, several subjects were not able to express the conditions in natural language at all (e.g. Subject 1, Transcripts-130), or only with much prompting from the investigator (e.g. Subject 8, Transcripts-143). Even those who managed to express the condition in natural language then failed to translate this into the formal structure of the programming language.

The transcripts reveal that most users did not have an adequate users' model of *objects*, *variables* and how *instantiation* works. Many could not detach object and variable names from their semantic meaning: they believed the system would not allow them to use "frivolous" variable names such as *Piglet* or *Pineapple*. They assumed the program would "recognise" variable names from the domain of family relations, such as *Brother* or *Father*. An appropriate user's model here requires that they strip terms used of the meaning attached to them in their general knowledge and experience (UW). Johnson-Laird's (1983) theory would predict that users have without training in formal knowledge have problems with this type of reasoning. Indeed, users had enormous difficulty to express real-world knowledge in formal logic; and there is also evidence of linguistic cueing throughout the transcripts, indicating that users are on the hunt for terms they recognise.

In retrospect, the discrepancy between the evidence in this study - that users did not have an appropriate users' model of the concepts and working of the programming language - and their apparently competent programming in the practice sessions can be explained. In the practice sessions, most users wrote their programs by *trial and error*; with the investigator correcting every incorrect statement immediately, users did not have a chance construct, test and modify a rule or query. Even if users were able to solve these task after trial and error, however, it is fair to state on the basis of that their users' models of Prolog had major omissions and misrepresentations.

10 MacWrite Study

The purpose of this study was to investigate the methods users tried to apply whilst *learning a new system* (a second word processing application). 17 users participated in this study, which employed another *constructive interaction* scenario: a co-investigator was paired with each user as a contrived co-learner to engage users in dialogue as they explored the system. The word processor is a Macintosh application, designed to facilitate access and encourage exploration by novice and casual users (as discussed in 4.2.2). The users in the main study already had experience of one word processing system (see Chapter 6). In order to determine to what extent their existing users' models of that system influenced their interaction with this new system, a comparison group of 19 users with no previous knowledge of word processing or computing was investigated as well.

10.1 Procedure

In the final study, users were asked to explore a word processor (MacWrite) in a one hour session. The *"observing users learning a system"* scenario was left for the last of the 5 studies, to see if the users in the main group would draw on models they had acquired during the six month course, and in particular of the word processing they had been trained in at the beginning of the course. A comparison with a group of users who were in the same situation as these users had been six months previously could be used to determine how their existing users' models their interaction with the system they were to explore. The investigator introduced the user to a "co-learner" (co-investigator) and asked them to work together to find out how to use the word processing system to edit and print a document. Like the scenario used in the spreadsheet study (see Chapter 7), this one is based on *constructive interaction* (Miyake, 1986) and devised to make *thinking aloud* more natural through dialogue. The idea is that when observing two people who jointly try to solve a problem, the investigator can capitalise on the fact that communication of knowledge takes place naturally between the two users.

The scenario is best described as *joint exploration*. A straightforward *constructive interaction* approach would have paired two users and asked them to explore the

system together. In view of the fact that all the other studies observed one user per session, it was decided to vary the approach and pair each user with a contrived co-learner (co-experimenter). This variation on constructive interaction also counteracts the possible danger of one user taking charge and dominating the interaction, in which case there would be little knowledge elicited from the less active user. The script for this scenario was as follows:

(1) The investigator suggested that the user should operate the keyboard, while the contrived co-learner should consult the manual whenever the user felt that s/he needed some specific information about the system.

(2) Users were briefly introduced to the operation of the mouse and the pull-down menus by the investigator.

(3) They were asked to edit a prepared document, which contained instructions about the tasks they had to perform (as shown in Table 25).

(4) As with the teach-back scenario in study two, the success or failure in eliciting users' models depends largely on the skill and intuition of the co-investigator: if the co-investigator is perceived to be knowledgeable about the system, the user will rely on the more expert "co-learner" to feed them the procedures required to solve the task, and not contribute very much to the interaction. In a bid to avoid such a situation arising, the co-investigators were briefed to be interested, but not overly helpful: they were not to respond too quickly to questions from the user, and always had to look up answers in the manual. When they had located the relevant text, they were – in the first instance – to give a vague answer, rather than reveal the exact procedure. (For instance, if the user could not work out how to change the margin, the co-investigator would say *"It's got something to do with the ruler"*). When a user made a mistake, the co-investigator would prompt him/her, by asking probing questions such as *"What went wrong there?"*.

Task	Description
1 Save file	Click on the File menu and select save
2 Change margins	Change the margins to 1.5 on the left-hand side and 6.5 on the right-hand side
3 Insert word	Insert a missing word
4 Delete chars	Delete characters
5 Delete words	Mark words with the mouse and delete them
6 Delete line	Delete a line
7 Double spacing	Change the spacing of the entire document to double-spaced
8 Single spacing	Change the spacing of the entire document to single-spaced
9 Centre paragraph	Centre one paragraph
10 Underline/embolden	Underline and embolden one word
11 Cut and paste	Move a paragraph by cutting and pasting
12 Print document	Print the document
13 Quit document	Leave the document without saving it
14 Create new document	Create your own document and type a letter
15 Save new document	Save your new document

Table 25: MacWrite tasks

10.2 Results

10.2.1 Results of the main group

17 users participated in the main group. The transcripts are provided in Transcripts 168-199, and Table 26 gives an overview of the performance on the tasks. Most users could not work out how to

- change the margins (16);
- delete words by marking them (16);
- double-space the text (15);
- centre a paragraph (17); and
- move a paragraph by cutting and pasting (15).

Observations surrounding these tasks are reported in more detail in the following subsections.

Change margins

Almost all users suggested FORMAT or clicked on the FORMAT menu straight away, and had to be directed to the ruler by the co-investigator reading the relevant instruction from the manual. Many selected the *Insert ruler* command on the FORMAT menu when the co-investigator mentioned that *"it has something to do with the ruler"*, even though there was a ruler already at the top of the document. When they had changed the left-hand margin, most users moved *the paragraph indent marker* instead of the *margin marker*. Even when they had moved the right-hand margin marker, they did not realise that the left-hand margin marker was still in its original place (e.g. Subject 12, Transcripts-187). The fact that only the first lines of the paragraphs were indented to 1.5 in., did not alert them to fact that they had moved the wrong marker, either.

Many users accidentally selected (and therefore highlighted) the ruler when they were trying to move the margin markers. Most of the users to whom this happened seemed very concerned, since they could not explain what had happened, and what effect highlighting the ruler had. In the majority of these cases, the co-investigator had to take charge and help users to recover, by suggesting that they clicked the cursor in the document to undo the marking of the ruler.

Double-spacing

Quite a few users did not know what *double spacing* meant. After it was explained by the co-investigator, one of them went straight for the ruler and selected the right icon (Subject 4, Transcripts-174), whereas another suggested to insert a blank line between every line of text (Subject 6, Transcripts-178). Most users opened the FORMAT menu, or verbally suggested that was where the command was likely to be. Once they arrived at ruler (most users had to be directed to the ruler by the co-investigator), half the users identified the double-spacing icon immediately, whereas the others went for the *6 lines/inch* box or one of the other two spacing icons.

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Total y
1 Save file	y	y	y	y	y	y	n	n	y	-	-	n	y	y	y	-	y	-	y	y	13
2 Change margins	n	n	n	n	n	n	n	n	n	-	-	n	y	n	n	-	n	-	n	n	1
3 Insert word	y	n	y	y	n	n	n	n	y	-	-	y	n	n	y	-	y	-	y	y	9
4 Delete chars	n	n	y	y	n	y	y	y	n	y	-	y	n	n	y	-	y	-	y	y	11
5 Delete words	n	n	n	n	n	n	n	n	n	n	-	n	n	n	n	-	n	-	y	n	1
6 Delete line	n	y	y	y	y	y	y	y	y	y	-	y	y	n	y	-	y	-	y	y	15
7 Double spacing	n	n	n	y	n	n	n	n	n	n	-	y	n	n	n	-	n	-	n	n	2
8 Single spacing	n	y	n	y	y	y	n	n	y	y	-	y	y	y	y	-	y	-	y	y	13
9 Centre paragraph	n	n	n	n	n	n	n	n	n	n	-	n	n	n	n	-	n	-	n	n	0
10 Underline/embold.	n	n	y	n	y	y	y	n	n	y	-	n	n	n	y	-	n	-	y	n	7
11 Cut and paste	n	y	n	n	n	n	n	n	n	n	-	n	n	n	n	-	n	-	n	n	1
12 Print document	y	n	n	n	y	n	y	n	-	n	-	y	n	n	y	-	y	-	y	n	7
13 Quit document	n	y	y	y	y	y	y	y	y	n	-	n	y	y	y	-	n	-	y	-	12
14 Create new doc.	y	n	n	n	y	y	n	n	y	n	-	n	y	-	y	-	n	-	n	-	6
15 Save new doc.	y	y	y	n	n	y	y	-	y	n	-	y	y	-	y	-	y	-	y	-	11
Tasks attempted	15	15	15	15	15	15	15	14	14	12	-	15	15	13	15	-	15	-	15	12	
Total y	5	6	7	7	7	8	6	3	7	4	-	7	7	3	10	-	7	-	10	5	

Table 26: Performance of main group on MacWrite tasks

Delete words (marking)

The instructions clearly stated that users should mark words with the mouse and then press the DELETE key. Most users could not achieve this because they could not work out how to mark a piece of text with the mouse. Most of them tried to mark the piece of text by clicking the cursor at the beginning and at the end (e.g. Subject 3, Transcripts-172). Some discovered by accident how dragging with the mouse would mark a piece of text (e.g. Subject 6, Transcripts-178). Many ignored the instructions and deleted the words by positioning the cursor at the end of the words and using the BACKSPACE key.

Centre paragraph

The instructions stated that users should insert a ruler at the top of the paragraph to be centred. Most users went straight to the FORMAT menu and selected *Insert ruler* without positioning the cursor first, thereby inserting the ruler where they did not want it. This started a sequence of hiding rulers, inserting another one (which had the effect that both rulers were displayed), trying to get rid of the unwanted ruler (the co-investigator would eventually suggest highlighting and deleting), the user centring the text below the ruler, and then deleting the ruler to get rid of it (e.g. Subject 5, Transcripts-178).

Cut-and-Paste

Only one user managed to apply the cut and paste operation without help. Users encountered a variety of problems (listed in order of frequency):

- they tried to select *Cut* before marking the paragraph;
- they forgot to position the cursor before selecting *Paste;*
- they were upset when the paragraph disappeared from the screen after cutting it, and convinced that something had gone wrong;
- they selected *Paste* before *Cut* and pasted the contents of the clipboard over the marked paragraph.

Additional Observations

(1) *Cursor/pointer confusion:* Many users in this group were clearly confused (at the least of the beginning of the session) by having a pointer as well as

a cursor. Even though they had been shown at the beginning of the session how to position the cursor by clicking the mouse, many positioned the pointer and started typing without clicking (e.g. Subject 6, Transcripts-178), and subsequently could not explain what had happened. Many users tried to move the cursor by "dragging" it with the pointer from its current position to the target position (e.g. Subject 9, Transcripts-184).

(2) *Marking/positioning cursor:* Many users were confused as to when they had to mark a piece of text before selecting an operation, and when it was sufficient to position the cursor somewhere in the text. When underlining or emboldening text, many users positioned the cursor at the beginning or in the middle of the word and selected the command.

(3) *Icons* vs. *names:* Most users showed a preference for selecting names (e.g. menu commands) when they were looking for commands, rather than pictographic commands. This could be observed particularly on the *change margins* and *double-spacing* tasks – most users would explore menus before considering the icons commands on the ruler. When they did so, it was often in response to prompting from the co-investigator, rather on their own initiative.

10.2.2 Results of the comparison group

19 users participated in the comparison group. The transcripts are provided in Transcripts 200-261, and Table 27 gives an overview of the performance on the tasks. Users in the comparison group took longer to complete the tasks (more than half did not complete the tasks in the one hour allocated). They encountered considerable problems with:

- change margins (all)
- double-spacing (all)
- centre paragraph (all)
- cut and paste (17).
- delete characters (13)

Change margins

Most users in this group explored the ruler as their first choice for this task, and guessed that the scale had something to do with it. Many correctly identified the paragraph indent and margin markers (some pointed out that they were in line with the text margins), but had problems working out how to move them (and often selecting the ruler in the process). A substantial minority opted for the FORMAT menu as their first choice, and a two users drew the analogy with a typewriter and suggested using the TAB key.

Double-spacing

Once users identified, or were directed to, the ruler for changing the spacing, they quickly identified the spacing icons. Since the ruler was not visible (they had scrolled down the text), most started to explore the menus and did not consider the ruler to start with.

Centre paragraph

Most users inserted a ruler as soon as they had read the instructions, without considering where it would be inserted. This started a cycle of hiding and inserting rulers: users removed rulers from the screen using the HIDE command, only to have them re-appear when they inserted another ruler. Most users would then try to make the ruler disappear, often discovering by accident that they could mark a ruler and delete it. Some users correctly inferred from marking and deleting text that this might be a way of removing (rather than just hiding) unwanted rulers. After they had centred the paragraph, however, most users wanted to remove the ruler from the screen, and proceeded to delete it rather than using HIDE. This resulting undoing the formatting they had just completed. Most users could not work out that, in order to have only one paragraph centred, they had to insert another ruler below the paragraph. Quite a few users tried to move the ruler they had inserted below the paragraph, rather than inserting another one.

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Total y
1 Save file	n	y	n	n	y	n	n	n	n	n	n	n	y	n	y	n	n	n	y	5
2 Change margins	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	0
3 Insert word	y	y	y	n	n	n	n	n	y	n	-	n	n	n	y	y	y	y	y	9
4 Delete chars	n	y	n	n	y	n	n	n	y	y	n	n	n	n	n	y	n	n	y	6
5 Delete words	n	n	n	n	y	n	n	y	y	n	n	y	n	y	y	y	n	n	n	7
6 Delete line	y	n	y	y	y	y	y	y	y	y	y	y	y	y	n	n	y	y	y	16
7 Double spacing	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	0
8 Single spacing	y	y	y	y	y	y	n	n	y	y	y	n	y	y	y	y	n	y	y	15
9 Centre paragraph	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	0
10 Underline/embold.	y	y	n	n	n	n	n	y	y	n	n	n	y	y	n	n	n	n	y	7
11 Cut and paste	n	n	n	n	n	n	n	n	n	y	n	n	y	n	n	n	n	n	n	2
12 Print document	n	y	y	y	n	-	n	y	n	n	n	n	n	y	n	y	n	n	n	6
13 Quit document	n	n	y	y	y	-	n	n	y	n	-	y	n	y	y	y	n	y	y	10
14 Create new doc.	-	n	y	-	-	-	-	n	-	-	-	y	y	y	n	y	-	-	-	5
15 Save new doc.	-	y	n	-	-	-	-	y	-	-	-	-	n	y	n	n	-	-	-	3
Tasks attempted	13	15	15	13	13	11	13	15	13	13	11	14	15	15	15	15	13	13	13	
Total y	4	7	6	4	6	2	1	5	7	4	2	4	6	8	5	7	2	4	7	

Table 27: Performance of comparison group on MacWrite tasks

Cut-and-paste

Quite a few users managed to loose the paragraph they had to move because they selected *Paste* after marking the paragraph, thereby pasting whatever was on the clipboard over it. Not one of the users could explain what had happened. Those users who did choose *Cut* worried when the paragraph disappeared, and/or forgot to click the mouse to position the cursor before selecting *Paste*, thereby re-inserting the paragraph at the same place where they had cut it from.

Delete characters

The most common error observed with this task was that users moved the pointer, but did not click the mouse to move the cursor, before pressing the BACKSPACE or DELETE key. As a result, they deleted the last character(s) of the word they had inserted in the previous exercise. Some users made life difficult for themselves by not moving the pointer after they had positioned the cursor – this meant they could not see whether the cursor was positioned correctly, and resulted in deleting the wrong character.

Additional observations

(1) *Naming of menu items:* Semantic cueing through the names of menu items seemed to affect users' selection of commands. Users would, for instance, opt for *align left* to change the left margin.

(2) *Cursor/pointer confusion:* Many users confused pointer and cursor to begin with, and tried to drag the cursor with the pointer to move it. This would often lead to them accidentally highlighting a piece of text, and discovering how to mark text in the process. With some users the cursor/pointer confusion persisted throughout the session, causing problems (e.g. Comparison-Subject 4 with *Cut-and-paste*, Transcripts-214).

10.3 Discussion and Conclusions

10.3.1 Effectiveness of the scenario

The scenario produced on average slightly less than two pages of verbal protocols for users in the main group, and more than 3 pages per user in the comparison

group. There was a constant dialogue between most users and their co-investigator. Users asked a great number of questions of the co-investigators. The fact that users and co-investigators were engaged in joint problem solving led users to perceive the co-investigator as an ally to whom they could not only turn to for information, but share their frustration and irritation with. In their attempts to formulate a joint problem-solving strategy, they verbalised their own goals and models when speculating about the reasons for the system's behaviour.

10.3.2 Evidence of users' models

A general conclusion for both groups of users is that the graphical user interface of this word processor, which is based on a typewriter metaphor (UW → DC → C+MC → UC) and uses icons, was less "intuitive" than is claimed by advocates of these approaches (see 4.2.2). Many elements of the graphical user interface were not recognised as representations of the functions users were looking for: none of the users recognised the indent markers as means of changing the margins, and when located, only a few the icons for spacing. Users had to consult the manual via the "co-learner" to identify these functions, even though they were in front of them. Most of the users in both groups were "thrashing around" during the session; even though this was their first hour of interaction with the system, it suggests claims that metaphor-based user interfaces can be learnt quickly by exploration and do not require instruction (e.g. by reading the manual) are somewhat over-optimistic. In the case of this particular system, there are points in favour of the basic metaphor chosen for the conceptual model (DC) - after all, we know from the results in Study 1 (see Chapter 6) that most users in this group described the word processor as a typewriter. The problem with the paragraph indent marker arise from the fact that the designer's view of the metaphor (D(UW)) is more elaborate then the users' (UW): users obviously know a typewriter, but there are clear indications that most of them had no representation of finer points such as the distinction between the margin marker and a paragraph indent marker.

Furthermore implementation in the system image (C+MC) here seems to cause problems: user do not recognise some of the icons and commands. In fact, users were often astonished when told which icon represents the functionality which they

were looking for. The concept of rulers which can be hidden, but still retain the formatting, creates exactly those invisible objects against which Tognazzini (1991, see 4.2.1) advises. The transcripts also show that not one of the users appreciated the difference between *hiding* and *deleting* rulers (which undoes the formatting commands contained on the ruler). Another example is the use of the *cut-and-paste* function. Users are left with no feedback as to what happens when they *cut* a paragraph - there is no visible representation of the scrapbook (a criticism also made by Tauber (1988)), so the paragraph just seems to disappear, which worries them greatly. They also have no indication as to the contents of the scrapbook when selecting *paste*, which results in the loss of the text to be moved when they select *paste* without having *cut* the paragraph first. With several users, this experience adversely affected a correct model they held: they suddenly tried to select formatting commands (such as boldfacing) without marking the piece of text first, because *"the last time we did that it deleted it"* (Subject 2, Transcripts-171).

One of the main aims of this particular study was to determine whether previous experience influenced the way in which users in the main group approached this particular system. There are two clear examples of users in the main group taking a different approach from the user in the comparison group. In Task 2 (*change margins*) 14 of the 15 users which attempted to solve the task without being directed by the co-experimenter opted for the FORMAT menu as their first choice, and only 1 user explored the ruler first. In the comparison, only 5 of the 17 users who attempted the task without help chose format, whereas 12 opted for the ruler. This different preference indicates that they were directed by their existing model of the previous word processor (UC1 → UC2), and thus less "open" to cues from the system image C+MC).

The second example of users in the main group being influenced by previous use of the system is the manner in which they handled *cursor control* and *marking* of text. Having a pointer and a cursor clearly confused users in the main group to start with, since they had previously only had a cursor. Most users tried to move the cursor by "dragging" it from its current position to the intended one (e.g. Subject 4, Transcripts 174). Many of them also moved the pointer without clicking to position the cursor, so text was inserted at the existing cursor position rather than the

previous one (e.g. Subject 6, Transcripts-177). Many users in the main group were unhappy with using the mouse, and it found especially difficult to use for marking. Some used the cursor keys when they could, which meant that fewer of them discovered by accident how to mark a piece of text when they tried to "drag" the cursor to positions it. Several users persisted in using cursor keys for small editing tasks despite the instructions in the text. Most users mastered cursor positioning after 2-3 small editing tasks, but moving the pointer without clicking resurfaced again for many in the *cut-and-paste* task; most users then recalled that they should have positioned the cursor first.

Fewer users in the comparison group had problems with *marking* and *deleting* words than users in the main group. The recordings demonstrate that users in the main group were trying to apply a model of marking from the previous word processor system, where a marker had to be inserted at the beginning and end of the piece of text. Consequently, they tried to mark text by "clicking" at the beginning and end of a piece of text to be marked, which in this application means they are just moving the pointer around (e.g. Subject 3, Transcripts 172). Many users in the main group ignored the instructions and persisted in using the less efficient but familiar methods of BACKSPACE or DELETE keys to delete whole words and lines of text.

Despite these specific difficulties due to existing models, users in main group overall performed better on their first encounter with this new system than users in the comparison group, for whom this was the first computer system. On average, users in the main group completed 6.4 of 14.4 attempted tasks. Users in the comparison group, for whom this was the first system, on average completed 4.8 out of 13.6 attempted tasks.

11 Conclusions

The purpose of this chapter is to present the conclusions of the research work presented in this thesis. Conclusions are drawn for three aspects of research on users' models in HCI:

(1) The methodological contribution of the thesis: methods for eliciting and describing users' models.

(2) The substantive contribution of the thesis: the evidence of users' models found, and implications for theories of users' models and their application.

(3) How the results of this work can contribute to an integrated and applicable body of HCI knowledge of users' models.

11.1 Methodological contribution

11.1.1 Eliciting users' models

One of the two main goals of the research described in this thesis was to explore the potential of different experimental scenarios for eliciting users' models. A range of scenarios was employed, ranging from the traditional experimental-style set-up in Study 1 to the *joint exploration* scenario employed in Study 5. Conclusions about cost and benefit of these scenarios are presented in this section.

The empirical work presented in this thesis developed and implemented five different scenarios for investigating users' models. The scenarios can be evaluated in terms of:

- resources required to implement a scenario;
- amount of data on users' models elicited;
- quality of data on users' models elicited;
- effort required to analyse users' models.

Experimental-style scenarios

The experimental-style scenarios employed in the studies reported in Chapters 6 (word processing 1), 8 (database) and 9 (Prolog) are developments based on experimental approaches in the empirical studies reviewed in Chapter 4.3. Users work through a set of tasks, which can be used to derive performance data (task completion, time taken to complete). At the same time, users are prompted by the investigator to verbalise their thoughts, producing verbal data which can be analysed for indications of users' models. Constantly prompting users to *think aloud* during completion of tasks (in the *observing users using a system* scenario employed in Study 1, see Chapter 6) turned out to be a highly artificial way of eliciting verbal data. The most natural dialogue in this scenario takes place when the investigator encourages users to verbalise their reasoning about *errors*. The performance analysis in the studies confirmed that user errors are systematic (i.e. that errors occur on particular tasks and for the same reasons), which should be a pointer to flaws in users' models. Verbal statements related to errors (why it happened, what users expected to happen instead) certainly are a rich source of information on users' models (UCs). This type of scenario is, therefore, suitable for eliciting verbal information illuminating "faulty" users' models (UCs) and their causes. It can be used to pinpoint those action which users have problems with, and help to establish the source of the problem (e.g. interpretation of objects or terms in the system image (C+MC)).

These scenarios are, however, not suitable for eliciting information in cases where users make few mistakes. Lack of errors on tasks in general cannot be taken as an indicator of an appropriate user's model (see 4.4.3). The results of the experimental-style scenarios here do indicate that carefully constructed *transfer tasks* (which require users to infer the action required) are most likely to identify cases where good performance on learnt tasks masks omissions or misrepresentations in the UC.

Competent users with appropriate models will also provide little verbal information under these circumstances. We may use an experimental-style scenario and performance data to identify highly competent users; in order to elicit "good" users' models (UCs), however, a scenario such as *teach-back* will have to be employed (see below).

Observing users predicting the behaviour of a system (Study 4 of the Prolog program, see Chapter 9) and *describing a system* (Study 7 involving the database system, see Chapter 8) are interesting variations on traditional observation of user-system interaction. Aspects of both scenarios could be combined to elicit and check specific aspects of users' models (UCs); techniques such as *what-if* questions and variations of the *cognitive jogthrough* (Rowley & Rhoades, 1992) could be employed as part of such scenarios to explore details of users' models. Such studies will require considerably more careful planning than a simple performance test and user observation. They do not, however, require more effort to run and analyse than a simple performance test, whilst providing the investigator with considerably more confidence that the UC held by users which pass these tests support user-system interaction adequately. Such scenarios could be usefully employed to check that users of safety-critical applications, for instance, have established a users' model (UC) which leads them to select the appropriate action for each circumstance, or to evaluate whether instruction and training in the use of a system has established a user's models which allows completion of a set of core tasks. It is important to recall that to give a reliable prediction of users' competence in using the system, users must have access to the system during such tests (see 5.1.3).

Constructive Interaction Scenarios

There can be little doubt that constructive interaction scenarios are more conducive to users verbalising their thoughts. In the two constructive interaction scenarios (the *teach-back* scenario used with the spreadsheet system in Study 2, see Chapter 7; and the *joint exploration* scenario employed with the MacWrite system in Study 5, see Chapter 10), users' communication of their thoughts and knowledge became part of a higher-level task (i.e. teaching another person about the system, or jointly completing a task). The style of conversation and users' body language indicate that users were more involved in this type of dialogue. The presence of a video camera means that users were, to some extent, still aware of being investigated, but the mere presence of another person sharing that situation seemed to make them more at ease, and more communicative. Thus, the elicitation of verbal data changes the *demand structure* of the interaction situation less than in the experimental-style

scenarios (see above). The two constructive interaction scenarios differed in three aspects:

(1) *Same* vs. *different level of knowledge:* In the teach-back scenario, the user is cast in a situation where s/he supposedly has more knowledge of the system than the "learner". The interaction is therefore led by the user, who carries the burden of decision-making and communicating knowledge. In the joint exploration scenario, users and "co-learners" are supposedly equally knowledgeable, and the responsibility for keeping communication going is shared.

(2) *Dependent* vs. *ally:* In the teach-back scenario, the user is responsible for driving the interaction. Most users were conscious of the fact that if they got into a situation from which they could not recover, they could not expect much help from the "learner" (co-investigator). In the joint exploration scenario, the burden of driving the interaction is shared: the user has control of the mouse and the keyboard, but the "co-learner" (co-investigator) had control of the repository of knowledge about the system (the manual). If users got into a situation from which they could not recover, there was still the option that the "co-learner" could think of other options, or find something in the manual.

(3) *Tightly structured* vs. *loosely structured:* In the teach-back scenario, the situation is only loosely structured - users have to define their own tasks (U(WT)) as well as work out how to perform them on the system (U(CT)). This clearly places an additional cognitive demand on users. Many users in the main group found it difficult to identify suitable tasks, and were looking to the "learner" to suggest what they should do with the system. This observation reveals interesting information about their UC of this system - namely that the (U(WT)) part is not particularly well developed. It is also clear that loosely structured scenarios allow users to avoid aspects of the system they do not know or feel uncertain about. The fact that users prefer some system tasks (U(CT)), and avoid others, reveals *some* information about their users' models (UCs), namely which parts of the system are represented in the UC to a degree which makes users

confident enough to tackle them. It does not, however, provide evidence about specific omissions or misrepresentations. In the joint exploration scenario, users were given a set of tasks to work through. This meant that users had to cover all parts of the system selected for investigation, and could concentrate on identifying the device operations (U(CT)) required to perform the tasks.

Users in the main group seemed to be more at ease, and behaving most naturally, in the joint exploration scenario. The co-investigator, who was perceived by most users to be an ally in problem-solving, was equally responsible for keeping the interaction going, and users did not have to worry about thinking of something else to teach the learner. Joint exploration seems a promising approach for eliciting knowledge from intermediate users.

Less experienced users can be expected to take a more passive role in joint exploration scenarios, since they lack the confidence and are desperately looking for guidance from the "co-learner" who has the manual. Users in the comparison group were less active, and constantly looked for guidance from the co-investigator. In this situation, a genuine co-learner might be more useful than a contrived one, since it can be expected that real co-learners will be less effective with the manual, and make suggestions which will turn out to be wrong.

When eliciting knowledge from more experienced users, the teach-back scenario should yield more insight than with intermediate or novice users: the users in the comparison group were reasonably confident about their interaction with the system. To experienced users, having to direct interaction is no problem, since they did not worry about "getting stuck", and were able to formulate relevant tasks for the learner to do. These users were keen to communicate knowledge to the "learner" throughout the interaction. They revealed much about their users' models by setting goals for the learner, and communicating what they judged to be essential knowledge of the system.

It has to be said that, whilst constructive interaction scenarios offer a great deal of potential for eliciting users' models, the cost in terms of preparation, execution and analysis is high. Contrived co-investigators have to be carefully trained, and all co-investigators employed in these studies found the sessions exhausting. Sessions

have to be recorded. Transcription and analysis are, as always with ethnographic methods, an extremely time-consuming and demanding exercise. What emerged during the analysis of the data collected in these studies is that comparisons of different statements (different actions from the same user or same actions from different users) can help to clarify the nature of a particular observation and identify the characteristic of a user's model which causes it. Unfortunately, like books, videotapes are linear artifacts: extracting comparative scenes from videotapes is very time-consuming and difficult. This has changed since the studies reported in this thesis were conducted: digital recordings and tools for indexing and annotation provide researchers with much improved facilities for detailed comparison and analysis of recordings. They also make it easier to have observations viewed and interpreted by a number of researchers. Multiple independent assessments would improve the reliability of results from such studies considerably.

11.1.2 Describing users' models

The transcripts (provided in Volume 2) contain not only the verbal data elicited in the studies, but descriptions of users' and (co-)investigators' non-verbal behaviour and system actions have been added. It is necessary to include this type of information in the transcripts to determine what may have triggered a particular user action, and to interpret users' statements as accurately as possible. Even with this information included in the transcripts, it was often necessary to return to a videotape and replay a situation (see above) when rating users' performance and identifying evidence of users' models. A first-pass analysis of the transcripts was conducted by the author, scoring users' performance following the criteria set out in 5.4.3. Systematic user errors were used as a guide for selecting those parts of the transcripts which have been analysed in more detail. Closer inspection of user behaviour surrounding errors yielded evidence about omissions and misrepresentations in part of users' models, as discussed for each study in Chapters 6-10. In addition, an attempt was made to elicit summary descriptions of the word processing system in Study 1 and the spreadsheet in Study 2, plus a comparison of spreadsheets and databases in Study 3. Those high-level verbal descriptions provided valuable additional information about users' models, and helped to illuminate behaviour observed (*viz.* the number of users drawing on the typewriter

metaphor (see Table 15); the impact of the visual structure of the system image of the spreadsheet (see Tables 18 and 19); and the change in model of the spreadsheet after introduction of the database (see Table 22)). Given that these simple, summary descriptions of the system are very easy to obtain, it seems worthwhile to elicit these almost as a matter of course when collecting data on users' models.

In retrospect, however, it seems that the design of the studies was biased towards eliciting verbal information from users about their models, so it is not surprising that the descriptions of users' models constructed from the data are mainly in this format. Alternative ways of describing users' models worth considering include drawings and diagrammatic representations, and schema-like representations.

Drawings and diagrammatic representations

Given the discussion above, it worth considering whether users should be encouraged to express their models other in than just verbal form. Whilst in all studies paper and pencil were on the desk on which the system was located, users were not encouraged to use it instead of, or in addition to, their verbal descriptions. Only one user spontaneously seized paper and pencil and started drawing a matrix when explaining the spreadsheet to the "learner" (Comparison Subject 6, Transcripts-78). Perhaps this is because users had access to the system in all studies, and preferred to point at, or demonstrate, a feature or function on the system itself. Gray's (1990) study is an example of how such drawings can be used to elicit and illustrate structural aspects of users' models. Payne (1990) suggests that diagrammatic representations could be used to illustrate the relation-structure of users' models - such as hierarchies of functions, or what is a function and what an attribute. The first-pass analysis of the data elicited in these studies does not particularly suggest any representations of structural aspects of the systems which would lend themselves to this type of description.

Diagrammatic representations were, however, useful to illustrate differences between models. The difference between the a user's and designer's model of can sometimes be illustrated in this manner (*viz.* the example of *reading* and *writing* files Figure 3 and Figure 4 in Chapter 6). Similarly, different sources of users' models, and how they influence the construction a user's model, is best most concisely described in this manner (see Figure 5 and Figure 6 in Chapter 8).

Schema-type representations

High-level descriptions of users' models might also be expressed as properties which can be added to an existing model. The descriptions of the word processor presented in Table 15 showed that half of the users described the word processor as a "Typewriter+". Figure 7 shows an example for describing a user model as an extension of an existing model, such as a metaphor. The functions, components and properties listed summarise all additions provided by users who described the system as a "Typewriter+" (see Table 15). In terms of Nielsen's taxonomy (see 4.1.4) this description is a *generic* users' model (R(UC)) prepared by the author. This particular notation represents an *instantiated* UC (of Subject 16, see Transcripts-19), whose description covered all additions except for those printed in italics. This demonstrates how such a notation can be used to compare generic and instantiated UCs, e.g. when we want to check to what extent the users' model intended by the designer (D(UC)) has been established after training and/or use of the system.

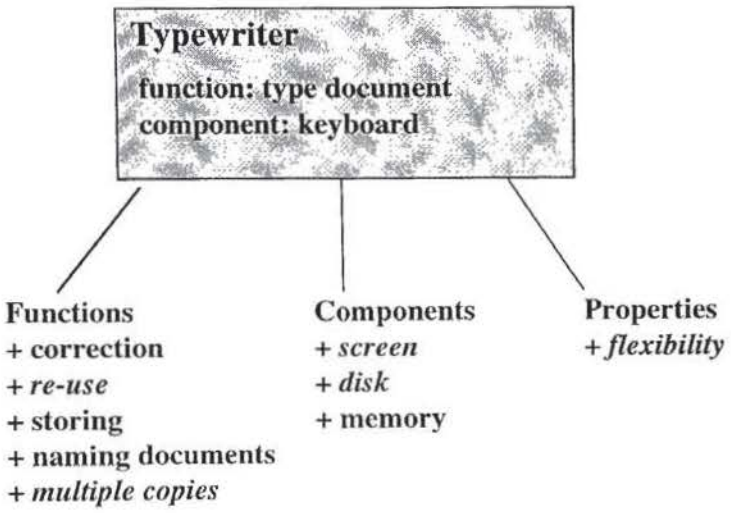


Figure 7: User's model (UC) of word processor as extension of typewriter

This type of notation can be particularly useful to describe users' models based on metaphors (UW), since several authors (Manktelow & Jones, 1987; Wozny 1989) emphasise that successful application of these requires that differences between the source and target models are identified by users. We can add "-" as well as "+" extensions to the above notation to denote features of the source model which do not

apply to the target model. This notation can therefore also be used to visualise the amount of *conceptual baggage* (Anderson et al., 1994) a metaphor carries (see 11.2.3)

Figure 7 represents a high-level description of a whole system (the word processor). The evidence collected in the studies shows, however, that misrepresentation often occur in detailed parts of the model. Schema-like descriptions can be generated for these more detailed aspects of users' models, such as attributes of the basic parameters text and numbers discussed in 7.3.2. Users in the two groups have different models of text and numbers which does affect the their performance on some tasks. Whilst users in the comparison group distinguish between text and numbers on the basis of *function*, users in the main group define them in terms of "face value", i.e. the elements they are composed of. Table 28 shows an example notation for representing these differences in users' models. This flaw in their user's model does not support entering of formulae or labels which contain numbers.

Parameter	**Type**	**Main Group**	**Comparison Group**
ENTRY		*defined by composition*	*defined by function*
	TEXT	- composed of letters	- label
	NUMBER	- composed of numbers	- data (integer, decimal) - formula

Table 28: User's models (UC) of text and numbers in spreadsheet

In a similar vein, the discussion of results from the database in 8.3.2 pointed out differences between the designer's model (DC, as communicated in the manual MC) of the data types and the users' model (UC). Half of the users did not distinguish between numbers and decimals, and saw binary data as a form of text. Tables 29 and 30 show schema-type representations of these two models.

Data	Type	
	TEXT	alphanumeric
	BINARY	binary choice
	DATE	01-JAN-1990
	NUMBER	integer
	DECIMAL	any decimal

Table 29: Designer's model (DC) of data types

Data	Type	
	TEXT	- alphanumeric - binary
	NUMBER	- integer - decimal
	DATE	- mixture TEXT/NUMBER

Table 30: Users' model (UC) of data types

These more detailed schema-like descriptions can be linked to the higher-level description in Figure 7 (e.g. by having a table for each function or component which has more detailed properties. Similarly, diagrammatic representations, such as the one in Figure 4, could be added to illuminate detailed aspects of users' models.

The discussion in this section demonstrates that models can be described at different levels and in different ways, and that there is no single best medium or notation for describing users' models. At this stage of research, it seems advisable to avoid the dangers of premature formalisation (Green et al., 1988). Indications are that a single method of description for users' models is unlikely to be appropriate. Users' models are not neatly structured and well-defined entities (see 11.2.1). As Payne (1991a) states:

> *"To understand mental models, a unitary description may not suffice. Rather, we need to understand the varied collection of tools in the bricoleur's bag." (p. 19)*

If users are opportunistic in their choice of the mental representations they employ, the researcher has to find appropriate descriptions for all of them. The influx of methods and tools from related disciplines in recent years has broadened the spectrum of both elicitation methods and descriptions which could be used by HCI researchers. Payne (1991b) suggested that conversation analysis might be applied to describe models, and it is worth considering techniques which have been employed by ethnographers interested in everyday cognition (Sperber, 1996).

11.2 HCI knowledge of users' models

The main aim of the research reported in this thesis was to develop HCI methods for investigating users' models. Even though the studies were not designed to evaluate any of the theories on users' models reviewed in 4.3, they have yielded evidence which can be added to existing HCI knowledge of users' models. This section summarises the evidence and discusses the implications for theory and design practice.

11.2.1 Structure and content of users' models

Textbook descriptions of users' models (e.g. Preece, 1994) and some of the theoretical contributions convey the impression that users' models are neat, compartmentalised entities. The evidence here is that real, instantiated users' models are anything but. The transcripts provide examples of every single one of Norman's (1983) observations (see 4.1.3):

- Instantiated users' models (UCs) are incomplete, easily forgotten, not clearly defined.
- Users do not "run" a mental simulation of the system in order to infer the correct action, they opt for *trial-and-error*.
- Even when they know more effective and efficient procedures, they prefer to stick with well-established ones.

As Erik Hollnagel once put it (personal communication, 1992), most users seem content with "muddling through" their interactions with the system. If this is true, it raises the question whether conceptual design, despite its intuitive appeal as a design approach, is doomed to fail in practice. Why invest time and effort in

designing a conceptual model (DC) and implementing a system image (C+MC) if users will not develop the envisaged users' model (D(UC)) anyway?[36] And what is the point of helping users to construct an appropriate users' model if they do not apply it when interacting with the system?

There are several reasons for persisting. First of all, it is worth emphasising again that the studies reported in this thesis do not constitute a test of conceptual design or any of the variations summarised in Table 10. The users' models (UCs) elicited are distributed, incomplete, incorrect, etc., but that does not mean that users will ignore design models (DCs) communicated through the system image (C+MC). With the exception of the last study (see Chapter 10), the systems investigated were not designed to communicate such models. In this situation, users draw on their existing knowledge and experience (UW), and try to find their own guiding principles (such as metaphors or analogies). This process is a haphazard one. There is clear evidence in these and other empirical studies that the system image (C+MC) has a some influence on users' models. Therefore, formulating an appropriate guiding principle (DC) and communicating it through the system image (C+MC) must stand some chance of "getting through" to the users' model (UC). The result may not be the ideal users' model envisaged by the designer (D(UC)), because users' existing knowledge, other cognitive processes or other factors (see below) may interfere with the construction process. But interacting with a system which communicates and reinforces the correct model should help users to get closer to that ideal than leaving them to find one from their own knowledge and experience (UW).

As to why users interacting with systems sometimes apply incorrect or inefficient procedures against better knowledge (behaviour which Norman (1983) describes as *"unscientific"* and *parsimonious*), we have to accept that the different phenomena which influence users' behaviour are not as neat and compartmentalised as they appear in psychology textbooks. Other cognitive mechanisms - such as memory - are part of the model construction process (Manktelow & Jones, 1987). And we have to consider that users' behaviour is not guided by cognition alone:

---

[36] *Viz.* also Tognazzini's (1991) quote in 4.2.1.

- User actions may be caused by perceptual cues and motor responses which bypass cognitive mechanisms altogether.
- Other factors, such as fatigue, can influence how well the cognitive machinery works.
- Like much of human behaviour, users' behaviour is opportunistic. Having an appropriate users' model is a means to an end, and if the end can seemingly be achieved faster and with less effort by other means - such as *trial-and-error* - those may be chosen in preference.

The fact that other mechanisms influence users' behaviour should not lead us to conclude that it is not worth trying to influence users' cognitive processes. Rather, HCI researchers wishing to support designers have to abandon their current compartmentalised model of these processes, identify mechanisms which can influence the model construction process, and specify how they influence the way in which models are used. In other words, the generic, general and analytic RU currently specified in the conceptual design process has to be made more specific and descriptive, and ultimately turned into DU in the form of guidelines. An example would be recommending that on-line tutorials and training, for instance, ought to be structured to allow for the limitations of working memory and provide opportunity for consolidation of users' models.

11.2.2 How users' models are constructed

As with structure and content of the users' models, the process by which they are constructed seems to be less clearly defined than the mappings in Table 10 suggest.

The evidence suggests users do not draw on one source exclusively for their users' models, and disregard all others. In the spreadsheet study (see Chapter 10), for instance, users in the main group were influenced by the system image (C+MC), but at the same time drew on knowledge of the previous system - with a completely different system image - in the manner of an analogy (UC1 → UC2). Users in the comparison group had no problem formulating a real-world task (U(WT)), but, instead of mapping this onto the functionality provided by the system (CT), continued to draw on their existing knowledge (UW).

It interesting to note how much users drew on existing knowledge and experience (UW), whereas most of the mappings suggest users' tasks (UT) as the prime source of design models (see Table 10). After analysis of the transcripts, the author would suggest that there are no clear boundaries between users' model of the task (UT) and their general knowledge and experience (UW). Was the typewriter metaphor employed by users in the first word processing study (see Chapter 6) a tool employed in the existing model of the physical U(WT), or part of their general knowledge? The answer would vary depending on how much experience users had with using a typewriter for producing documents. Conceptual design seems to have been right in hedging its bets by taking both UT and UW (plus mental mechanisms RU, see above) as sources for the design model (DC).

The evidence in the studies demonstrates that a well-formulated model of the real-world task (U(WT)) is - as Payne et al.'s (1990) YSS states - an essential part of an appropriate users' model. Researchers often make the tacit assumption that users are experts at their tasks and therefore have a well-formulated U(WT) that system functions (U(CT)) can be mapped onto. A major lesson to be drawn from the studies is that, if a conceptual design exercise is to be successful, such assumptions about users' knowledge should be replaced by accurate assessment of user knowledge. Researchers' and designers' models of the users' tasks (R(UT)and D(UT)) and their general knowledge and experience (R(UW) and D(UW)) are no substitute for the real thing - UT and UW. It is not only important to accept that users' minds are not blank slates for designers to write on (Carroll & Rosson, 1987), but that effort is required to establish what is on the slate before we start adding to it. If users' models of the real-world tasks (U(WT)) are not sufficiently well developed, training users in procedures (CT) will not lead to formation of a model. Given that introduction of a computer-based tool to a real-world task often changes the nature of the real-world task (WT), extending training and documentation (MC) to provide explicit support for the construction of U(WT) as well as U(CT) would be beneficial even when users have appropriate models of the current real-world task.

11.2.3 Analogies, metaphors and users' models

Evidence in the studies suggests that users try to apply their existing knowledge and experience (UW) to user-system interaction (*viz.* the typwriter metaphor in Study 1, Chapter 6; the analogy with the previous word processing system in the Study 5, Chapter 10). Analogies and metaphors are special cases of existing knowledge and experience, in that they should allow transfer of significant chunks of an existing model into a users' model (UC). Yet, analogies and metaphors as instructional design models have met with limited success (see discussion in 4.4.3), and the evidence in these studies suggests that users who are adapting self-selected analogies (UC1 → UC2) and metaphors (UW → UC) may be hindered as much as helped in the process of constructing an appropriate user's model. This evidence should, however, not lead us to discount metaphors and analogies as sources for design models (DC). There is a plethora of evidence of analogies and metaphors being successfully applied to facilitate the construction of new models in many other domains (e.g. Lakoff & Johnson, 1980; Gentner & Stevens, 1983).

Rather, the somewhat slapdash manner in which metaphors and analogies are currently selected and applied in user interface design (see 4.2.2) needs to be replaced by a more painstaking process. Firstly, source models should be identified by empirical means, rather than just assuming that users have particular models in a particular format. In practice, this requires discussing source models which designers think are suitable with users, to establish that those parts relied on in the mapping are "present and correct". It is also possible to identify models by discussing the application domain or envisaged system functionality with (a representative group of) users, and see which models this evokes[37]. Secondly, the amount of *conceptual baggage* (Anderson et al., 1994) carried by a source model has to be determined. Source models which cover *too much* beyond the intended UC introduce an undesirable cognitive overhead, which is even more detrimental than models which do not cover enough of the intended UC. Thirdly, the implementation of the design model in the system image (C+MC) must represent differences between the source model (UW or UC1) and the design model (DC) to

[37] This approach has been used by the author in a recent - as yet unpublished - case study.

support adaptation of the model cued - for instance, using a representation similar to that in Figure 7. Finally, linguistic consistency between the source and target models needs to be considered (see below).

11.2.4 Linguistic cueing

The transcripts contain plenty of evidence of linguistic cueing. When users are looking for the a command, they will often select one containing a term used in the instructions (e.g. *align left* when they have been told to *set the left margin marker* in Study 5). This suggests that linguistic elements of the system image, such as commands, and explanations in help and error messages, ought to be selected with great care. Users also tend to chose terms they are familiar with in preference to others, even in cases where logical inference would rule out a command (e.g. the use of *move* to write a paragraph to a separate file, see 6.3.2). These observations confirm that linguistic format of elements and instructions determines users' inference, as predicted by Manktelow & Jones (1987). A related implication is that are that terms employed in the system image (C+MC) which are not consistent with the meaning of that term in UW or UT will cause problems. The process of transferring parts of source models into a user's model is likely to be impaired if the meaning of terms employed in the system image differs from the meaning stored a user's mental lexicon (Aitchison, 1994). Thus, ensuring linguistic compatibility is particularly important in the implementation of system images (C+MC) which try to draw on analogies or metaphors (see 11.2.3), but the same principle applies when designing an application based on an existing real-world task (U(WT)). Users' models may not be entirely determined by propositional semantics (Johnson-Laird, 1983 - see Table 4), but the linguistic element clearly plays a large part in the model-building process. Linguistic consistency between source and target models can be achieved by choosing linguistic elements of the system image (C+MC) to match their counterparts in the source model. In practice, this means that:

(1) designers should not create new terms in the system image (C+MC) for elements which are already defined in (UW) or (UT);

(2) designer should only employ terms which are already defined in (UW) or (UT) in the system image (C+MC) if the meaning is consistent[38].

These two suggestions indicate that linguistic engineering of a system image (C+MC) is likely to be a complex and time-consuming exercise. Nevertheless, similar linguistic approaches have become increasingly popular as part of requirements analysis, to ensure that terms in the computer system are consistent with the terms employed by users (Liebenau & Backhouse, 1990)[39]. Such an approach may appear more time-consuming and painstaking, but it is a fairly straightforward technique which could form part of eliciting users' knowledge about tasks (UT) and background (UW).

11.2.5 Basic parameters and users' models

The studies have provided some examples of users' models which contained faulty representations of basic parameters used by the application, evidence of the detrimental impact of faults in these basic building blocks on user performance on certain tasks. This evidence confirms a central part of YSS. Payne et al. (1990) state that the construction of basic parameters is a substantial cognitive overhead, and failure to establish them means semantic mapping between real world task (WT) and system task (CT) cannot be properly established.

The representation of text and numbers in the spreadsheet study (see 7.2.3 and Table 28) provides the most compelling example of such a problem. Users' failure to distinguish between data types is an extension of this problem in the database application (see discussion in 8.2.3 and Tables 29 and 30).

Users' lack of understanding of Boolean operators in the database study is another example. This adds to similar results by Borgman (1986) and Marchionini (1989); thus, we can be fairly sure that the problems in the use of abstract concepts predicted

---

[38] Figure 4 shows an example of users applying the UW meaning to the terms READ and WRITE rather than the DC one.

[39] This process is not necessarily one-way: the analysis process often reveals inconsistent use of terms *within* the organisation, e.g. that different users attribute a different meaning to the same term. This has to be resolved by agreeing a consistent set of definitions which has to be learnt by users (i.e. their model of UT has to be changed) as well as implemented in the system.

by Johnson-Laird's (1983) theory applies to computer users without training in maths and logic. This is not surprising, but gives some cause for concern about the way in which applications which do incorporate such functions are currently used by many. User training which - as in this case - teaches use of retrieval functions by example on a series of tasks, without explaining Boolean logic "glosses over" a difficult but important concept. Users who do not establish an appropriate model of this basic parameter will not be able to use the retrieval function properly, and are less likely to realise when they have specified search criteria incorrectly. Since retrieval is a core function of this application, the effort required to establish a correct UC of Boolean logic seems a small price to pay compared to a lifetime of sub-optimal and error-prone use of a core function.

In the Prolog programming exercise (see Chapter 9), users demonstrated a similarly incomplete and faulty UC of objects and variables, which explains their poor performance on the programming tasks in the session, especially transfer tasks. This is another example of competent performance on learnt tasks (in their coursework) masking a mis-representation of a basic parameter.

In conclusion, the evidence on lack of understanding of basic parameters raises the question as to how effective pure task-action models (UT $\rightarrow$ UC) can be. It seems that a surrogate model (U $\rightarrow$ UC), at least of basic parameters, is required for competent use of some applications. If users do not have such a model from their general knowledge and experience (UW) - and most do not - it has to be established as part of training.

11.3 Towards a theory of users' models

The conclusions of the review of existing knowledge of users' models in 4.4 identified 4 main items for the research agenda:

- unifying terminology;
- integration of existing general knowledge;
- building HCI theory of users' models;
- developing HCI methods for investigating users' models.

The research reported in this thesis mainly addressed the final point, but has in the process made a contribution to the other 3 areas as well. The purpose of this concluding section is to summarise these contributions and outline further research required to (a) progress HCI knowledge of users' models, and (b) make it applicable by designers of user interfaces.

11.3.1 Developing HCI methods for investigating users' models

The motivation for the work described in this thesis was to obtain examples of specific, instantiated users' models in action. This is important to further both research and application of users' models in interface design.

From a research point of view, the author felt that is was necessary to create an empirical counterpoint to the existing proliferation of proposed models (see Appendix 1) and mapping relationships (see Table 10); the diverging opinions as to the structure and content of users' models can only be resolved by empirical investigation. The evidence of users' models found (see 11.2) provides confirmation for parts of existing theories as well as adding some new aspects (see 11.3.4).

From an application point of view, it is essential to provide instantiated UCs for the benefit of designers, who, presented with lots of abstract theory and no examples, struggle to see how this theory might be applied to the design process.

In order to provide the empirical evidence, HCI methods for eliciting and describing users' models have to be developed. The work reported in this thesis represents a first step towards a suite of scenarios and associated techniques for elicitation (see 11.1.1), which could be used by researchers and designers seeking to elicit users' models of systems (UC) which have been constructed as a result of user-system interaction and/or instruction. Following the discussion in 11.2.3, it also seems necessary to consider a set of methods which would be suitable for eliciting users' models of tasks (U(WT)) and their general knowledge and experience (UW) as *input* for a *conceptual design*-based process. Apart from identifying existing models which might function as a source for a users' model, it is essential to capture the *terms* in which users describe them. The author also feels that existing HCI methods for *task analysis* could provide such input; reviewing the output of different methods of task analysis, to see how they could be adapted for this purpose, would therefore not

only help to make conceptual design more concrete, but also be a step towards integration with other strands of HCI research.

11.3.2 Unifying terminology

In order to overcome the problems resulting from the *"confused and confusing"* (Payne, 1992) terminology surrounding users' models (see 4.4.1), the author has adapted Nielsen's (1990) taxonomy of models in HCI (see 4.1.4) to provide a taxonomy of users' models and related models (see Appendix 1). The notation has been applied to:

- distinguish the different claims made by existing theoretical contributions in (4.2);
- assess the evidence from existing empirical contributions (in 4.3); and
- discuss evidence of users' models generated in the studies (in Chapters 6-10).

The author feels that development and application of the taxonomy and notation has been an extremely worthwhile exercise. If adopted by further theoretical and empirical work to describe models proposed and elicited, it would further the integration of HCI knowledge of users' models. It could especially help to compare different design approaches for user interfaces aimed at supporting users' models (in the manner described in Table 10). The use of such a notation would also help to formulate clear guidelines on applying HCI knowledge of users' models; formulation of such guidelines would help designers, especially if they were integrated with other HCI tools and methods (see above).

11.3.3 Integration of existing general knowledge

The conclusion of the review of evidence of users' models (see 11.2) has to be that existing general knowledge of mental representations and models has much to offer and should be integrated with HCI knowledge. Only one attempt has been made (by Manktelow & Jones, 1987) to apply one of the theories - Johnson-Laird's (1983) theory of mental models - to HCI, and this attempt has been all but ignored by their fellow researchers and by designers. More recently, Payne (1990, 1992) has indicated that Johnson-Laird's theory actually encompasses many of the claims and findings on users' models in HCI. A consequence of this ignorance is that the linguistic

dimension of users' models (UC) and system images (C+MC) has been very much neglected.

Considering the evidence on metaphors (see 11.2.3) and linguistic cueing (see 11.2.4), the author would suggest that successful conceptual design relies on perspiration more than inspiration. Careful stocktaking (to establish existing knowledge and its linguistic format) and engineering the details of the system image (C+MC) (to support cueing and integration) may be more important than a creative brainstorm to create a metaphor which explains the system in one grand sweep. Even when a good metaphor has been found, it needs to supported by the all elements of the system image (C+MC) with which the user interacts. This applies not only to the linguistic elements, but the visual ones, too. Visual elements do indeed make a strong impression on users and influence what they remember about the system (UCs), but this potential is currently not realised as much as it could be. Like the linguistic elements, visual components of most system images (C+MC) are currently not as carefully selected and tested as they should be. Existing knowledge of physical and mental representations (see 3.1) can provide a useful framework for distinguishing different types of representations, and the function they can perform; it would be useful to incorporate this knowledge in the training of user interface designers.

11.3.4 Building HCI theories of users' models

The aim of the work reported in this thesis was to make a contribution towards establishing an integrated and applicable body of HCI knowledge of users' models. The work has focused on two very basic research needs - to provide evidence of real, instantiated users' models (UC), and developing methods for eliciting and describing such evidence. Having addressed two of their 11 basic research needs, how far are we from establishing a theory of users' models as Carroll & Olson (1988) envisaged it?

> *" ... a theory of users' models needs to be embedded in a model of a full-blown cognitive system, one that has problem solving and decision-making processes that are sufficient to initiate the model "runs", collect the results, and decide on external action." p. 59*

The answer has to be: (a) we have a very long way to go; and (b) if and when we get there, the theory may look very different from what they envisaged. Firstly, the integration of relevant existing knowledge of human cognition has not been tackled (see 11.3.3); secondly, it is now obvious that a valid theory of users' models cannot be limited to cognition (see 11.2.1).

Even integration of knowledge generated under the banner of HCI has, until now, not taken place. The author believes that there are two main paths to be pursued in changing this state of affairs: adoption of a unified terminology and framework (see 11.3.2), and grounding research in data on users' models. Much of the divergence of the theoretical contributions is due to the fact that researchers keep producing conceptualisations of users' models (R(UCs)), rather then examining users' model (UCs) themselves. Empirical research needs to be guided by theory, but theory needs to be grounded in reality - i.e. data - to provide valid guidance for action in the real world.

To provide support for users' models, researchers and designers need to focus on working with users. Conceptual design may have originated in a volume titled *User-Centered System Design* (Norman & Draper, 1986), but most researchers and designers seem to prefer working with representations of users and their models (R(UCs), D(UW), etc.). The problem is that the conceptualisations we have are not a sufficiently accurate representations of how users think and behave. It seems likely that they could be improved to some extent by testing and integrating existing knowledge (see above). At the same time, progress can be made by grounding both research and application in data - an approach which befits an applied discipline like HCI.

Researchers could evaluate their own theories against data, such as those provided in Volume 2 of the thesis, in the traditional science manner by specifying UCs users should have according to their theory, and using performance, errors and verbal statements as indicators. Payne et al.'s (1990) YSS, for instance, predicts that users should be able to infer optimal procedures if appropriate models of the real-world task (U(WT)) and the system (U(CT)) exist, and a mapping between them has been established.

Following the conclusions on the state of the discipline (see 2.3) a more promising approach may be to formulate theories of users' models from data, using grounded theory methods (Glaser & Strauss, 1967). Grounded theory methods can be used to construct low-level models (paradigms) by classifying phenonema into categories, and specifying interactions between them. The resulting model is then tested against all observations in which the phenomenon occurs. If the model holds true in all observations, it is accepted as a building block; if not, the paradigm has to be revised. Interactions between paradigms are specified at a higher level (storyline) and tested again; once a storyline has been verified, it is equivalent to a theory. An added advantage of theory-building through grounded theory is that existing knowledge in both quantitative and qualitative format can be integrated in this process. Grounded theory would allow integration of parts of existing theories which can be validated against data, and thus support the integration of existing HCI and general knowledge into a theory of users' models.

On a practical level, designers could contribute to this process by grounding design models (DCs) on users' models of tasks (UT) and the real world (UW), and carefully engineering the system image (C+MC) to communicate that model. Systems developed in this manner are artifacts which would allow researchers to put the idea of conceptual design to test.

References

Aaronson, A. & Carroll, J. M. (1987):

The Answer is the Question: a Protocol Study of Intelligent Help. *Behaviour And Information Technology*, **6**, 393-402.

Abelson, H., Sussman G. J. & Sussman, J (1985):

Structure and Interpretation of Computer Programs. Cambridge, MA: MIT Press.

Aitchison, J. (1994):

Words in the Mind: an introduction to the mental lexicon. 2nd Edition. Oxford: Blackwell.

Anderson, B.; Smyth, M.; Knott, R. P.; Bergan, M.; Began, J. & Alty, J. L. (1994):

Minimising Conceptual Baggage: Making Choices about Metaphor. In In Cockton, G., Draper, S. W. & Weir, G. R. S.[Eds.]: *People and Computers IX - Proceedings of the Ninth Conference of the British Computer Society HCI Specialist Group*. Univ. of Glasgow, Sept. 1994. Cambridge: CUP.

Angell, I. O. & Smithson, S. (1991):

Information Systems Management: Opportunities and Risks. London: Macmillan.

Apple Computer Inc. (1987):

Apple Human Interface Guidelines: The Apple Desktop Interface. Reading, MA: Addison-Wesley.

Baecker, R. M.; Grudin, J., Buxton, W. A. S. & Greenberg, S. [Eds.] (1995):

Readings in Human-Computer Interaction: Toward the Year 2000. 2nd Edition. San Francisco, CA: Morgan Kaufman.

Bayman, P. & Mayer, R. E. (1984):

Instructional Manipulation of Users' Mental Models for Electronic Calculators. *International Journal Of Man-Machine Studies*, **20**, 189-199.

Bellotti, V. (1988):

Implications of Current Design Practice for the Use of HCI Techniques. In

Winder, R. & Jones D. M. [Eds.]: *People and Computers IV. Proceedings of the Fourth Conference of the British Computer Society HCI Specialist Group.* Univ. of Manchester, 5-9 Sept. 1988. Cambridge: CUP.

Booth, P. A. (1991):

Modelling the User: User-System Errors and Predictive Grammars. In Weir, G. R. S. & Alty, J. L. [Eds.]: *Human-Computer Interaction and Complex Systems.* London: Academic Press.

Borgman, C. L. (1986):

The User's Mental Model of an Information Retrieval System: an Experiment on a Prototype Online Catalog. *International Journal of Man-Machine Studies,* **24,** 47-64.

Briggs, P. (1988):

What we know and what we need to know: the user model versus the user's model in human-computer interaction. *Behaviour and Information Technology,* **7,** 431-442.

Buxton, B. (1995)

Absorbing and Squeezing Out: On Sponges and Ubiquitous Media. *Proceedings of the International Broadcasting Symposium,* November 13-16, Tokyo, 91-96.

Card, S. & Moran, T. P. (1986):

User Technology: From Pointing to Pondering. *Proceedings of the ACM Conference on History of Personal Workstations,* Palo Alto, 183-198.

Card, S., Moran, T. P. & Newell, A. (1983):

The Psychology of Human-Computer Interaction. Hillsdale, NJ: LEA.

Carroll, J. M. & Campbell, R. L. (1989):

Artifacts as Psychological Theories: the Case of Human-Computer Interaction. *Behaviour and Information Technology,* **8,** 247-256.

Carroll, J. M. & Aaronson, A. P. (1988):

Learning by doing with Simulated Intelligent Help. *Communications of the ACM,* **31,** 1064-1079.

Carroll, J. M. & Olson, J. R. (1988):

Mental Models in Human-Computer Interaction. In Helander, M. [Ed.]: *Handbook of Human-Computer Interaction*. Amsterdam: Elsevier.

Carroll, J. M., Smith-Kerker, P. L., Ford, J. R. & Masur-Rimetz, S. A. (1987/88):

The Minimal Manual. *Human-Computer Interaction*, **3**, 123-153.

Carroll, J. M. & Rosson, M. B. (1987):

Paradox of the Active User. In Carroll, J. M. [Ed.]: *Interfacing thought: Cognitive Aspects of Human-Computer Interaction*. Cambridge, MA: MIT Press.

Carroll, J. M. & Campbell, R. L. (1986):

Softening up Hard Science: reply to Newell and Card. *Human Computer Interaction*, **2**, 227-249.

Carroll, J. M. ; Mack, R. L.; Lewis, C. H.; Grischkowsky, N. L. & Robertson, S. R. (1985):

Exploring a Word Processor. *Human Computer Interaction*, **1**, 283-307.

Carroll, J. M. & Mack, R. L. (1985):

Metaphor, Computer Systems, and Active Learning. *International Journal of Man-Machine Studies*, **22**, 39-57.

Carroll, J. M. & Mack, R. L. (1984):

Learning to use a Word Processor: by doing, by thinking, and by knowing. In Thomas, J. C. & Schneider, M. [Eds.]: Human Factors in Computing Systems. Norwood, NJ: Ablex.

Collins, A. & Gentner, D. (1987):

How People Construct Mental Models. In Holland, D. & Quinn, N. [Eds.]: *Cultural Models of Language and Thought*. Cambridge: Cambridge University Press.

Craik, K. (1943):

The Nature of Explanation. Cambridge: Cambridge University Press.

Czaja. S.; Hammond, K. ; Blascovich, J. J. & Swede, H. (1986):

Learning to use a Word Processing System as a Function of Training Strategy. *Behaviour And Information Technology*, **5**, 203-216.

Diaper, D. [Ed.] (1989):

Task Analysis for Human-Computer Interaction. Chichester: Ellis Horwood.

DiSessa, A. (1986):

Models of Computation. In Norman D. A. & Draper S. W. [Eds.]:*User-Centered System Design: New Perspectives in Human-Computer Interaction*. Hillsdale, NJ: LEA.

Dowell, J. & Long, J. (1989):

Towards a Conception for an Engineering Discipline of Human Factors. *Ergonomics*, **32**, 1513-1535.

Frese, M., Albrecht, K.; Altmann, A.; Lang, L.; Papstein, P. V.; Peyerl, R.; Pruemper, J.; Schulte-Goecking, H.; Wankmueller, I. & Wendel, R. (1988):

The Effects of an Active Development of the Mental Model in the Training Process: Experimental Results in a Word Processing System. *Behaviour and Information Technology*, **7**, 295-304.

Gaines, B. R. & Shaw M. L. G. (1986):

From Timesharing to the Sixth Generation: the Development of Human-Computer Interaction – Part 1. *International Journal of Man-Machine Studies*, **24**, 1-27.

Gentner, D. (1983):

Structure-Mapping: A Theoretical Framework for Analogy. Cognitive Science, **7**, 155-170.

Gentner, S. & Stevens, A. L. [Eds.] (1983):

Mental Models. Hillsdale, NJ: Erlbaum.

Glaser, B. & Strauss, A. (1967):

The Discovery of Grounded Theory. Chicago: Aldine.

Goodman, N. (1984):

Languages of Art: An Approach to a Theory of Symbols. 2nd Edition. Indianapolis, IN: Hackett.

Gray, S. H. (1990):

Using Protocol Analyses and Drawings to Study Mental Model Construction

during Hypertext Navigation. *International Journal of Human-Computer Interaction*, **2**, 359-377.

Green, T. R. G. (1990):

The Cognitive Dimension of Viscosity: a Sticky Problem for HCI. In Diaper, D. et al. [Eds.]: *Human-Computer Interaction – Interact'90. Proceedings of the IFIP TC 13 Third International Conference on Human-Computer Interaction*, Cambridge, 27-31 August, 1990, pp. 79-86. Amsterdam: North-Holland.

Green, T. R. G.; Schiele, F. & Payne, S. J. (1988):

Formalisable Models of User Knowledge in Human-Computer Interaction. In Van Der Veer, G.; Green, T. R. G.; Hoc, J. M. & Murray, D. M. [Eds.]: *Working with Computers: Theory versus Outcome*. London: Academic Press.

Grudin, J. (1992):

Utility and Usability: Research Issues and Development Contexts. *Interacting with Computers*, **4**, 209-217.

Grudin, J. (1991):

Interactive systems: Bridging the Gaps between Developers and Users. *IEEE Computer*, **24** (4), 59-69.

Halasz, F. G. & Moran, T. P. (1982):

Analogy Considered Harmful. *Proceedings of the Conference on Human Factors in Computing Systems*. Gaithersburg, MD, March 1982, 383-386.

Johnson, J. (1987):

How Faithfully Should the Electronic Office Simulate the Real One? *SIGCHI Bulletin*, **19** (2), 21-25.

Johnson-Laird, P. N. & Wason, P. C. (1970):

Insight into a Logical Relation. *Quarterly Journal Of Experimental Psychology*, **22**, 49-61.

Johnson-Laird, P. N. (1983):

Mental Models. Cambridge: Cambridge University Press.

Kieras, D. E. & Polson, P. G. (1985):

An Approach to the Formal Analysis of User Complexity. *International Journal of Man-Machine Studies*, **22**, 365-394.

Kieras, D. E. & Bovair, S. (1984):

The Role of a Mental Model in Learning to Operate a Device. *Cognitive Science*, **8**, 255-273.

Kuckenberg, M. (1989):

Die Entstehung von Sprache und Schrift: Ein kulturgeschichtlicher Überblick. Köln: Dumont.

Lakoff, G. & Johnson, M. (1980):

Metaphors We Live By. Chicago, IL: Unversity of Chicago Press.

Larkin, J. & Simon, H. A. (1987)

Why a Diagram is (sometimes) worth Ten Thousand Words. *Cognitive Science*, **11**, 65-100.

Liebenau, J. & Backhouse, J. (1990):

Understanding Information: An Introduction. London: Macmillan.

Long, J. & Dowell, J. (1989):

Conceptions of the Discipline of HCI: Craft, Applied Science, and Engineering. In Sutcliffe, A. & Macaulay, L. [Eds.]: *People and Computer V. Proceedings of the Fifth Conference of the British Computer Society Human-Computer Interaction Specialist Group*, Univ. of Nottingham, 5-8 Sept. 1989. Cambridge: CUP.

Manktelow, K. & Jones, J. (1987):

Principles from the Psychology of Thinking and Mental Models. In: Gardiner, M. M. & Christie, B. [Eds.]: *Applying Cognitive Psychology to User-Interface Design*. Chichester: Wiley.

Marchionini, G. (1989):

Making the Transition from Print to Electronic Encyclopaedias: Adaptation of Mental Models. *International Journal of Man-Machine Studies*, **30**, 591-618.

Mayes, J. T., Draper, S. W. McGregor, A. M. & Oatley, K. (1988):

Information Flow in a User Interface: the Effect of Experience and Context on

the recall of MacWrite screens. In Jones, D. M. & Winder, R. L. [Eds.]: *People and Computers IV*. Cambridge: Cambridge University Press.

Miyake, N. (1986):

Constructive Interaction and the Iterative Process of Understanding. *Cognitive Science*, **10**, 151-77

Monk, A. F. & Wright, P. C. (1991):

Observations and Inventions: New Approaches to the Study of Human-Computer Interaction. *Interacting with Computers*, **3**, 204-216.

Monk, A. (1986):

Mode Errors: a User-Centred Analysis and Some Preventative Measures Using Key-Contingent Sound. *International Journal of Man-Machine Studies*, **24**, 313-327.

Moran, T. P. (1983):

Getting into a System: External-Internal Task Mapping Analysis. In Janda, A. [Ed.]: *Human Factors in Computing Systems I.* Proceedings of the CHI'83 Conference held in Boston, MA, 12-15 Dec. 1983 (ACM SIGCHI). Amsterdam: North Holland.

Moran, T. (1981)

The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems. . *International Journal of Man-Machine Studies*, **15**, 3-50.

Newman, W. & Lamming, M (1995):

Interactive System Design. Wokingham: Addison-Wesley.

Newell, A. & Card, S. (1986):

Straightening out Softening up: a response to Carroll and Campbell. *Human Computer Interaction*, **2**, 251-267.

Newell, A. & Card, S. K. (1985):

The Prospects for Psychological Science in Human-Computer Interaction. *Human-Computer Interaction*, **1**, 209-242.

Nielsen, J. (1990):

A Meta-Model for Interacting with Computers. *Interacting with Computers*, **2**, 147-160.

Norman, D. A. (1988):

The Psychology of Everyday Things. New York, NY: Basic Books.

Norman, D. A. (1986)

Cognitive Engineering. In Norman D. A. & Draper S. W. [Eds.]:*User-Centered System Design: New perspectives in human-computer interaction*. Hillsdale, NJ: LEA.

Norman, D. A. (1983):

Some Observations on Mental Models. In Gentner, D. A. & Stevens, A. A. [Eds.] *Mental models*. Hillsdale, NJ: Erlbaum.

Norman, D. A. & Draper, S. W. [Eds.] (1986):

User-Centered System Design. Hillsdale, NJ: LEA.

O'Malley, C. E.; Draper, S. W. & Riley, M. (1984):

Constructive Interaction: a Method for Studying User-Computer-User Interaction. *Tech. Rep. C-015*. San Diego, CA: University of California, Institute for Cognitive Science.

Olson, J. R. & Olson, G. M. (1990)

The Growth of Cognitive Modeling in Human-Computer Interaction since GOMS. *Human-Computer Interaction*, **5**, 221-265.

Paivio, A. (1986):

Mental Representations: a Dual Coding Approach. New York, NY: Oxford University Press.

Payne, S. J. (1992):

On Mental Models and Cognitive Artefacts. In: Rogers, Y., Rutherford, A. & Bibby, P. [Eds.]: *Models in the Mind: Theory, Perspective & Application*. London: Academic Press.

Payne, S. J. (1991a):

A Descriptive Study of Mental Models: Understanding Users' Understanding. *Behaviour and Information Technology*, **10**, 3-21.

Payne, S. J. (1991b):

Display-Based Action at the User Interface. *International Journal of Man-Machine Studies*, **35**, 275-289.

Payne, S. J. , Squibb, H. R. & Howes, A. (1990):

The Nature of Device Models: The Yoked State Space Hypothesis and Some Experiments with Text Editors. *Human-Computer Interaction*, **5**, 415-444.

Payne, S. J. (1988):

Methods and Mental Models in Theories of Cognitive Skill. In: Self, J.: *Artificial Intelligence and Human Learning: Intelligent Computer-aided Instruction*. London: Chapman & Hall.

Payne, S. J. & Green, T. R. G. (1986):

Task-Action Grammars: a Model of the Mental Representation of Task Languages. *Human-Computer Interaction*, **2**, 93-134.

Preece, J., Rogers, Y., Sharp, H., Benyon, D. Holland, S. & Carey, T. (1994):

Human-Computer Interaction. Wokingham: Addison-Wesley.

Rogers, Y., Rutherford, A. & Bibby, P. [Eds.] (1992):

Models in the Mind: Theory, Perspective and Application. London: Academic Press.

Rowley, D. E. & Rhoades D. G. (1992):

The Cognitive Jogthrough: A Fast-Paced User Interface Evaluation Procedure. In Bauersfield, P. et al. [Eds.]: *Proceedings of CHI'92 - Conference on Human Factors in Computer Systems*. New York: Association of Computing Machinery.

Sasse, M. A. (1992):

Users' Models of Computer systems. In Rogers, Y., Rutherford, A. & Bibby, P. [Eds.]: *Models in the Mind: Theory, Perspective & Application*. London: Academic Press.

Sasse, M. A. (1991):

How to T(r)ap Users' Mental Models. In Tauber, M. J. & Ackermann, D. [Eds.]: *Mental Models and Human-Computer Interaction* 2.. Amsterdam: Elsevier.

Sasse, M. A., Johnson, G. I. & Briggs, P. (1988):

Introducing Word Processing to Novice Users: a Study of 'Procedural' and 'Conceptual' Approaches. In Megaw, E. D. [Ed.]: *Contemporary Ergonomics 1988: Proceedings of the Ergonomics Society's 1988 Annual Conference*, Manchester, England, 11-15 April 1988, 426-431.

Sasse, M. A. (1986):

Design and Evaluation of a Training Program for Word Processor Use: an Experimental Study. M.Sc. Thesis (unpublished), MRC/ESRC Social and Applied Psychology Unit, University of Sheffield.

Sperber, D. (1996)

Explaining Culture: A Naturalistic Approach. Oxford: Blackwell.

Strauss. A. & Corbin, J. (1990):

Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Newbury Park: Sage.

Suchman, L. (1987):

Plans and Situated Actions. Cambridge: Cambridge University Press.

Tauber, M. (1988):

On Mental Models and the User Interface. In Van Der Veer, G.; Green, T. R. G.; Hoc, J. M. & Murray, D. M. [Eds.]: *Working with Computers: Theory versus Outcome*. London: Academic Press.

Tognazzini, B. (1991):

TOG on Interface. Reading, MA: Addison-Wesley.

Van der Veer, G. C. & Wijk, R. (1991):

Teaching a Spreadsheet Application – Visual-Spatial Metaphors in Relation to Spatial Ability, and the Effect on Mental Models. In Tauber, M. & J. Gorny, P. [Eds.]: *Visualisation in Human-Computer Interaction*. Berlin: Springer.

Van der Veer, G. C. & Felt, F. A. M (1988):

Development of Mental Models of an Office System: A Field Study on an Introductory Course. In Van der Veer, G. C. & Mulder, B. [Eds.]: *Human-Computer Interaction – Psychonomic Aspects.* Heidelberg: Springer.

Wason, P. C. & Johnson-Laird, P. N. (1972):

Psychology of Reasoning: Structure and Content. Cambridge, MA: Harvard University Press.

Winograd, T. & Flores, F. (1986):

Understanding Computers and Cognition: a new Foundation for Design. Norwood, NJ: Ablex.

Wozny, L. A. (1989):

The Application of Metaphor, Analogy and Conceptual Models in Computer Systems. *Interacting with Computers*, **1**, 273-283.

Young, R. M. (1983):

Surrogates and Mappings: two Kinds of Conceptual Models for Interactive Devices. In Gentner, D. & Stevens, A. L. [Eds.]: *Mental models.* Hillsdale, NJ: Erlbaum.

Young, R. M. (1981):

The Machine Inside the Machine: User's Models of Pocket Calculators. *International Journal Of Man-Machine Studies*, **15**, 51-85.

Appendix 1: Classification of models

1 Models of the system

(1.1) **User's model (UC)**

An individual user's internalised representation of a specific computer system. The model is therefore instantiated. It is dynamic since it is assumed to change with user-system interaction, and likely to be distributed rather than structural for most users.

(1.1.1) **Designer's model of users' model (D(UC))**

An internalised or externalised structural, generic representation of the user's model, which is probably descriptive rather than analytic, specific rather than general, and (like all conceptualisations) static. Norman (1986) defined UC to cover both UC and D(UC), but a distinction between the generic and the instantiated model should be made.

(1.1.2) **Researcher's conceptualisation of users' model (R(UC))**

An externalised, structural, generic model of the user's model, which is static, general rather than specific, analytic rather than descriptive.

(1.2) **Design model (DC)**

An externalised, general design model of a specific system, produced by the designer. This will normally be structural, descriptive and static.

(1.2.1) **Researcher's conceptualisation of the design model (R(DC))**

The researcher's view of the conceptual model may be different from the designer's model. It is likely to be externalised, structural, generic, static, and analytical rather than descriptive.

(1.3) **System image (C+MC)**[40]

The way in which the design model is presented to the user, through the user interface, documentation, instruction material, error and help facilities. It contains of parts of the system, which is an expression of the design model, rather than a model in itself; models may, however, be provided as part of the accompanying materials and documentation (M).

(1.4) **Conceptual Model (UC+DC)**

Norman (1986) states that UC and DC form the conceptual model, without elaborating its characteristics or functions further. Many authors still use the term this term instead of design model (DC).

2 Models of Tasks

(2.1) **User's Task Model (UT)**

An internalised model of a task to be performed. This is an instantiated, specific, descriptive, dynamic model of the task. Whether the model is structural or distributed will to depend on the complexity of the task and user's level of expertise. We can expect users with similar training and experience of the task to have similar models.

(2.1.1) **Designer's Model of Users' Task D(UT)**

The designer's generic model of the UT, which can be internalised or externalised. The model is likely to be specific, descriptive and static. Whilst approaches such as conceptual design prescribe that the design model should be based on the user's model of the task, there is a danger that the designer may start from their own model of the task (DT), rather than model the task from the user's viewpoint (D(UT)).

---

[40] Nielsen (1990) assigns the label MC for system image. Since M is for Manual and other documentation, it could be argued that it does not cover undocumented parts so of the system which the user interacts with, such as the screen display. C+MC expresses Norman's (1986) notion of system image more precisely.

(2.1.2) Researcher's Conceptualisation of Users' Task (R(UT))

The analysis of the users' task can be conducted using a formalised method or tool (task analysis), and the resulting model will be externalised using some notation. Depending on the method or tool used, the model may be descriptive or analytic. Task analysis methods would claim that they model the users' model of the task, but if they are applied to the task as a purely analytical exercise, without verification by users, the result will be an R(T) rather than an R(UT). Task analysis methods aim to produce generic and general models, and are unlikely to reproduce the idiosyncratic and distributed features (including gains and inconsistencies) of an individual user's task model.

(2.2) User's Model of Real-World Task (U(WT))[41]

This is the the user's model of the task the user wants to perform in the real-world, i.e. the primary task. This is an internalised, specific, instantiated, descriptive and dynamic model of the task. It is related to the concept of *goal space* of Payne et al. (1990), which is a conceptualisation of this model.

(2.2.1) Goal space (R(U(WT)))

Researcher's conceptualisation of how users represent a real-world task, which differs from U(WT) in that it is externalised, generic, analytic and static. As with (R(UT), depending on the method used to derive this model, the result may be an R(WT) rather than an R(U(WT)).

(2.3) User's Model of System Task (U(CT))

The user's model of the operations required to be performed on the system to complete U(WT). This is therefore an enabling task. As a user's model, it will be internalised, specific, instantiated, descriptive and dynamic model; whether it is structural or distributed will depend on the user's level of expertise with the task.

---

[41] YSS is sometimes referred to as a model of the user. It is a model of the user in that it is a conceptualisation of how users represent tasks, but at the core is the task representation.

(2.3.1) Device space (R(U(CT)))

Researcher's conceptualisation of how users represent operations required to complete U(WT). It is an externalised, generic, analytic and static version of U(CT). Some methods used to analyse system tasks are R(CT)s rather than R(U(CTs)).

3 Models of the world

(3.1) User's Existing Knowledge (UW)

This is the background knowledge an individual user brings to the task, and which may influence the way in which they interpret information about the system and interact with it. This is a highly individual, internalised model which differs from UT in that it is general rather than specific. It is likely to be distributed and dynamic.

(3.1.1) Designer's Model of Users' Knowledge (D(UW))

Designers are likely to form an internalised generic model of the subset of users' background knowledge and experience which may be relevant to the task and system. This model is likely to be distributed and descriptive. As with the designer's model of users' tasks (D(UT)), there is a possibility that the designer's own model of the world (DW) might be used instead of the user's (D(UW)).

(3.1.2) Researcher's Conceptualisation of Users' Knowledge (R(UW))

This would differ from D(UW) in that it is likely to be externalised and analytic.

4 Models of the user

(4.1) Researcher's Model of Users (RU)

Research in psychology and related disciplines has produced a number of theories and models of the working of the human perception, memory, cognition and kinesthesia. These models are externalised, structural, generic, general, analytic and static.

(4.2) **Designer's model of users (DU)**

Designers might be using parts of RU, but are unlikely to be familiar with RU in its entirety. To follow Tognazzini's (1991) recommendations, they might consult a researcher to select or interpret a relevant part of RU for them. If this is not the case, they might draw on a their own, internalised version of RU, which is likely to be distributed, generic, and descriptive.

5 Mappings

(M1) **Conceptual Design (D(UT)+D(UW)+R(U) → DC → UC)**

The design model should be based on the users' task, and existing knowledge and experience. When constructing the design model, the designer should also incorporate knowledge about human perception, memory, cognition and motoric skills. The resulting design model is expressed in the system image; the user constructs a user's model which corresponds to the design model as a result of user-system interaction, and maybe training and instruction (D(UT) +D(UW) + R(U) → DC → C+MC → UC).

(M2) **Analogy (UC1 → UC2)**

Mapping an existing user's model of one computer system onto a new system. This requires, naturally, that the user holds a similar model which can act as source of the mapping. The design model and system image would incorporate the existing user's model. The mapping would be expressed in the system image, and the user would develop a new model through interacting with the system (D(UC1) → DC → C+MC → UC2). Alternatively, UC1 may be evoked and amended through instruction. Wozny (1989) suggests that with increasing knowledge about computing, more users will be able to use analogies when learning a new system.

(M3) **Metaphor (UW → UC)**

Mapping an existing model from user's knowledge onto the a new system. This requires identification of a suitable model, incorporating it in the design model, and communicating it via the system image. In interaction with the

system image, the user then develops a new model (D(UW) → DC → C+MC → UC).

(M4) **Surrogate models (C → UC)**

Users who hold a mental analogue of the system they are interacting with can predict system behaviour accurately, since they "run" a simulation of the system itself when using that model. These internalised, instantiated and specific models may be acquired through prolonged and regular interaction with the system. DiSessa (1986) suggests that users develop a task-action mapping of the system to start with, then develop a sense for the structure underlying the functions, and finally discover the true amount of functionality (DC → MC+C; (UT)+(UW)+(MC+C) → UC1; UC1+(MC+C) → UC2).

(M5) **Task-action mapping (UT → UC)**

Reduced models which account for the functionality of a system by reference to the real-world task the user is performing are called *task-action* mapping models (Young, 1983). Task-action mappings model a mental representation of a direct connection between a task and the user actions required to perform that task. In order to construct such a mapping the designer would select a minimum set of task-action relationships (the *core* of the model), and channel the remainder of the functionality through these core mappings (D(UT) → DC → C+MC → UC).

(M6) **Formal task-based mapping (R(UT)+R(U) → UC)**

Tauber (1988) proposes that design models should be based on formal models of the user's task; models of human cognition which are based on theory and validated by experiments should be added to the process (D(R(UT)+R(U)) → DC → C+MC → UC).

(M7) **Semantic mapping (R(U(WT) → U(CT)) → UC)**

The mapping relationship proposed by YSS (Payne et al., 1990) states that the U(WT) is mapped onto U(CT) via function called semantic mapping

(R(U(WT) → U(CT)) → UC). Semantic mapping is a proposed cognitive mechanism – a general RU – applied to a specific problem.

Appendix 2: Summary Principles from psychology of thinking and mental models for user interfaces design (Manktelow & Jones, 1987)

1. Thinking is not based on general inference rules, so competence on one task will not necessarily generalize to others of the same logical structure.

2. Performance required can be low (that is, routine, response-based) or high (that is, flexible, decision-based).

   2.1 If the performance required is low, then performance will depend on (a) heuristic processes, or (b) direct memories.

   2.2 If the performance required is high, then performance will depend on (a) existing schemata, or (b) new schemata.

3. Heuristic processes are automatic response biases generated by attention to salient task features (STFs). STFs are principally perceptual or linguistic and involve the natural meaning of these cues.

   3.1 If the STFs provide a clear task structure, performance will be facilitated.

   3.2 If the STFs are non-functional, performance will be impaired.

   3.3 If the natural and the required uses of the STFs are different, performance will be impaired.

4. Direct memories are automatic routines retrieved from the user's memory. Direct memories, existing schemata, and new schemata can be with respect to (a) the system, or (b) the task. Use of direct memories, existing or new schemata is a function of the user's familiarity with (a) the system, or (b) the task. Familiarity is a function of (a) assessment of the user's background, (b) the user's self-assessment, and (c) level of use.

   4.1 Highly familiar material cues direct memories

   4.2 Highly unfamiliar material cues heuristic processes.

5. Existing schemata are automatic transferable thinking procedures from the user's general knowledge.

   5.1 Material at a level of general familiarity will evoke existing schemata.

   5.2 Existing schemata can be evoked by verbal examples (analogies).

   5.3 If no familiar examples are available, new schemata are required.

   5.4 Induction of new schemata requires practice.

6. Verbal instructions do not in themselves inculcate thinking skills.

   6.1 If new schemata are required, verbal feedback can be used for their evaluation.

   6.2 Verbal material can be used to evoke existing schemata or direct memories.

7. If when an analogy or metaphor is first introduced in a dialogue on information is present to identify

   7.1 those task domains in which it use is appropriate, and

   7.2 those task domains in which use is inappropriate

   then the power of the analogy or metaphor will be reduced.

8. If when an analogy or metaphor is first introduced in a dialogue no information is present to identify

   8.1 those structures and functional relations that are essential, and

   8.2 those structures and functional relations that are incidental to the analogy or metaphor

   then the power of the analogy or metaphor will be reduced.

9. If no information is present in a dialogue when a structure or functional relation in the system or subsystem departs from what is conventionally the case [in the analogy or metaphor], then the power of the analogy or metaphor will become a negative power for each occurrence of that case.

10. If the sentences used in a dialogue contain 'marked' words, then they are read slower and less accurately than any root equivalents.

11. If the order of recall or the order of user actions is important, and the order of presentation of words in a dialogue is antagonistic to that order, then the probability of comprehending the dialogue will be reduced.

12. If the sentences used in a dialogue contain negatives, then they are read slower and less accurately than comparable sentences phrased in terms of affirmative.

13. If the sentences used in a dialogue are open to more than one interpretation, then they are read slower and less accurately than sentences that are open to just one interpretation.

14. If the sentences used in a dialogue contain referents that are not referred to, or introduced elsewhere in the dialogue, then the probability of constructing a single, coherent and plausible model will be reduced.

15. If the properties and relations ascribed to the referents of a sentence used in a dialogue are incompatible with, and contradict those that are ascribed elsewhere in the dialogue, then the probability of constructing a single, coherent and plausible model will be reduced.

16. If the user is a novice user, and if (s)he has not completed an outline tutorial programme, then the probability of constructing a single, coherent and plausible mental model will be reduced.

17. If the item of information to be conveyed by the dialogue is left implicit, or embedded within a sentence that is verbose, then the probability of identifying that information as a separate item will be reduced.

18. If the referent common to two separate items of information is designated in the two items by two different names, then the probability of identifying the referent as common to both will be reduced.

19. If the number of referents, or the number of properties and relations ascribed to the referents used in a dialogue exceed the capacity available in working memory, then the probability of considering all the relevant referents, or relevant properties and relations, will be reduced.

20. If sentences 1, 2, and 3 of a new screen of information do not explicitly relate that screen to the previous screen of information, then the probability of comprehending the dialogue will be reduced.

21. If the consequences of the possible user responses to any screen of information are not given explicitly, in terms of the information to be encountered on the next screen, the probability of comprehending the dialogue will be reduced.

22. If the items of any screen of information are independent of one another then the probability of comprehending the dialogue will be reduced.

23. If the scoring between a system's dominant functional characteristics and those of other systems is low, then the probablility of the user making the correct assumptions about the dominant functional characteristics of each of the major features and operations of the system is reduced.

24. If the probability of the user making the correct assumptions about the dominant functional characteristics of each of a system's major features and operations is low, then the probability of 'capture' will be increased.

Appendix 3: Initial interviews with subjects

Subject 1 (Greg - male, 38, Physical Education)

Q: Did you have any experience with computers before you came on this course?

S: Only as a keyboard - just typing in data.

Q: Did you ever play computer games?

S: No.

Q: Why did you come on this course?

S: It's absolutely essential to know how to use them today. You need to type a dissertation, do statistics, and you have to use a computer for all these.

Subject 2 (Steve - male, 19, Physical Education)

Q: Did you have any experience with computers before you came on this course?

S: No.

Q: Did you ever play computer games?

S: No.

Q: Why did you come on this course?

S: I thought it would be important to know about them. I want to have my own business later on, running a sports centre or something like that, and you should really use a computer to do the bookings and for accounts.

Subject 3 (Fiona - female, 19, German)

Q: Did you have any experience with computers before you came on this course?

S: No.

Q: Did you ever play computer games?

A: No - when I was 12 or 13 we did something at school, playing games on a computer, but I don't remember any of it.

Q: Why did you come on this course?

S: It thought it would be interesting - it's not exactly what I thought it would be - I thought it was going to be learning about programming - but it's important to have used computers today. Learning a language isn't sufficient to get a job today, you've got to have more of a background.

Subject 4 (David - male, 18, Geography)

Q: Did you have any experience with computers before you came on this course?

S: Not at all. We had a Spectrum to play games.

Q: Did you ever play computer games?

S: Space Invaders, that sort of thing, nothing eduational.

Q: Why did you come on this course?

S: I thought I should know something about computers. Our group really missed out completely at school. Now, a lot of junior schools have computers, but they didn't then. It's important if you want a job, they're bound to have computers.

Subject 5 (Tim - male, 19, Geography)

Q: Did you have any experience with computers before you came on this course?

S: No.

Q: Did you ever play computer games?

S: Yes, arcade games and space invaders and that sort of stuff.

Q: Why did you come on this course?

S: Well, you really need it today if you want to go into management or teaching, most things really. So I thought I'd try it here first, rather than make a fool of myself in my first job.

Subject 6 (Michael - male, 19, Music)

Q: Did you have any experience with computers before you came on this course?

S: We've got one in the Music department, for compositions and things. But I've never used anything else.

Q: Did you ever play computer games?

S: No.

Q: Why did you come on this course?

S: You can see that even in music there are more and more [computers] around. And if you want to go into teaching, you need to be able to do word processing and use other programs.

Subject 7 (Jonathan - male, 19, Geography)

Q: Did you have any experience with computers before you came on this course?

S: No - we had a computer studies course in my 3rd year at school, but it was all punch cards and thinks like that, we never got to see a computer. I haven't got the faintest idea what that had to do with what we are doing now.

Q: Did you ever play computer games?

S: I've played arcade games, but I've never used a home computer to play games.

Q: Why did you come on this course?

S: I thought that I was a bit of a computer illiterate. Your job opportunities widen once you've got experience with a computer.

Subject 8 (Caroline - female, 20, German)

Q: Did you have any experience with computers before you came on this course?

S: No.

Q: Did you ever play computer games?

S: No, I've never been interested in that.

Q: Why did you come on this course?

S: I think because we're in a computer age, it's just something you ought to have. I've always had a phobia of computers, but they are supposed to be useful, so I thought I'd try and see if I can manage. Anything you do these days, you've got to be able to use computers, in accountancy or admin. We have to write essays on a word processor soon.

Subject 9 (Peter - male, 18, Geography)

Q: Did you have any experience with computers before you came on this course?

S: No - not at all.

Q: Did you ever play computer games?

S: No.

Q: Why did you come on this course?

S: I thought it would be good in terms of employment afterwards. Also, they seem to be everywhere these days and I felt a bit out of it, not having a clue. It's frustrating, there seems to be all these "experts" around, and most of them aren't exactly geniuses, either. It would be useful later, even just be able to do word-processing.

Subject 10 (Cath - female, 20, Geography)

Q: Did you have any experience with computers before you came on this course?

S: I have used a computer at home before, but nothing like this.

Q: What did you use it for?

S: We bought this home computer, and went through the book to some extent, but mainly it was games after that. I have used a basic word processor on it.

Q: Did you ever play computer games?

S: Yes.

Q: Why did you come on this course?

S: Because computers interest me, and I really want to learn something about them. I don't have a clue what I want to do when I graduate, but I know people who have a geography degree and did some computing, and it's really helped them to find a job.

Subject 11 (Katherine - female, 19, Music)

Q: Did you have any experience with computers before you came on this course?

S: Well, I think it was one or two years at school, learning BASIC and things. And we did some drawing on the screen - turtle graphics.

Q: Did you ever play computer games?

S: I've got a Spectrum at home, which I have used just for games, Space Invaders and adventure games.

Q: Why did you come on this course?

S: It's very useful, I don't know very much about them. It comes in useful with Music as well. And with doing jobs later.

Subject 12 (Sharon - female, 20, Politics)

Q: Did you have any experience with computers before you came on this course?

S: Well, not really - I worked for a year and typed a bit then. I worked at a bank it was just correcting things.

Q: Did you ever play computer games?

S: My brother's got a small home computer - shooting men out of the sky.

Q: Why did you come on this course?

S: Because I know nothing about it. And I read that it would be good for when I leave, to find a job. And I when I was working with them, I was always frightened, so I thought I had to overcome this fear.

Subject 13 (Judy - female, 19, General Honours)

Q: Did you have any experience with computers before you came on this course?

S: Just to play games, and to draw graphs with it, in the 6th form - we had about 3 lessons. It was in Maths, but there were six or seven computers between 40 students, so all we did get round to is pressing buttons.

Q: Did you ever play computer games?

S: Space Invaders and that sort of thing, on a Spectrum at home. And a tutorial for o'level chemistry and French.

Q: Why did you come on this course?

S: 'Cos it looked interesting, and we were told that we should learn word processing. There was this guy who talked at us for an hour, and it left everybody confused. I'm scared of computers I thought it would be a good idea to learn something. I'd have to do it sometime, and I thought now would be a good time.

Subject 14 (Jane - female, 20 Music)

Q: Did you have any experience with computers before you came on this course?

S: Yes, for o'level, but it was all theory.

Q: What did you do?

S: We had to write a program in BASIC - mostly for storing files and recalling the information.

Q: Did you ever play computer games?

S: No.

Q: Why did you come on this course?

S: I'd like to use it for doing my dissertation and that sort of thing, word processing especially. I think I probably have to use them later on ... I'd like to teach ... not necessarily for music teaching, but in schools in general they will be used a lot

Subject 15 (Mark - male, 18, Physical Education)

Q: Did you have any experience with computers before you came on this course?

S: No. We had a computer to play games on.

Q: Did you ever play computer games?

S: Yes, but I can't remember what they were, Space Invaders.

Q: Why did you come on this course?

S: I'm quite interested in computers, I thought it would be useful in the future, I want to go into management, sports management.

Subject 16 (Angela - female, 20, General Honours)

Q: Did you have any experience with computers before you came on this course?

S: No.

Q: Did you ever play computer games?

S: No, never.

Q: Why did you come on this course?

S: I just wanted some experience, find out what you can do. I'm sure it would be useful for anything I might do after my degree.

Subject 17 (Liz - female, 18, English)

Q: Did you have any experience with computers before you came on this course?

S: We did a little bit of word processing at school, but not a lot. Just a couple to hours, to show what it can do.

Q: Did you ever play computer games?

S: No.

Q: Why did you come on this course?

S: I have always steered clear of computers, but it's so important now, you've got to know what they do, otherwise you can't do anything these days.

Subject 18 (Sarah - female, 19, French)

Q: *Did you have any experience with computers before you came on this course?*

S: No, not at all.

Q: *Did you ever play computer games?*

S: No.

Q: *Why did you come on this course?*

A: Well, computers are a thing of the future, so I wanted to learn about them. I thought I'd have to at some time ... I just wanted to learn more 'cos you hear so much about them, find out how they are used.

Subject 19 (Claire - female, 20, Politics)

Q: *Did you have any experience with computers before you came on this course?*

S: My dad's got one at home, so I've done a bit of typing and playing games on it.

Q: *Did you ever play computer games?*

S: A bit, but it gets boring very quickly.

Q: *Why did you come on this course?*

S: Well, you really should know something about them for most things - they're anywhere these days, especially in the office. I would like to go into journalism and you certainly have to be able to use them there.

Subject 20 (Rosemary - female, 18, Music)

Q: *Did you have any experience with computers before you came on this course?*

S: No.

Q: *Did you ever play computer games?*

S: No - I did once, when I was about eight, I think.

Q: *Why did you come on this course?*

S: I thought it would be useful. I haven't got any specific ideas now, but it might turn out to be useful later. And felt I ought to learn something about computers.

Appendix 4: Prolog programme (study 4)

```
% person/2 read:
% person(Person, Sex).
person(annie,female).
person(eric, male).
person(jenny, female).
person(martin, male).
person(zandra, female).
person(norman, male).
person(carol, female).
person(rupert, male).
person(harry, male).
person(james, male).
person(kathy, female).
person(peter, male).
person(rosie, female).
person(sophie, female).

% parent/2 read:
% parent(Parent, Child).
parent(norman, harry).
parent(norman, james).
parent(norman, kathy).
parent(jenny, harry).
parent(jenny, james).
parent(jenny, kathy).
parent(martin, peter).
parent(martin, rosie).
parent(carol, peter).
parent(carol, rosie).
parent(zandra, sophie).
parent(rupert, sophie).
parent(annie, jenny).
parent(annie, martin).
parent(annie, zandra).
parent(eric, jenny).
parent(eric, martin).
parent(eric, zandra).

father(Father, Child):-
```

person(Father, male),
parent(Father, Child).